Escuela
Politécnica
Superior

Recreación del escenario de una película en Unreal Engine 4

Grado en Ingeniería Multimedia

Trabajo Fin de Grado

Autor:
Manuel Moreno Ñíguez

Tutor/es:
Mireia Luisa Sempere Tortosa

Universitat d'Alacant
Universidad de Alicante

Septiembre 2018

Justificación y objetivo

Durante la carrera se han abarcado diferentes aspectos en relación con los contenidos multimedia: programación, diseño gráfico y 3D, gestión de proyectos, etc. Yo voy a realizar una escena 3D, por la que moverte e interactuar, simulando lo más fielmente posible la casa de la película *Don´t breathe* (*No respires*, en castellano).

Para ello voy a encargarme de modelar todos los objetos 3D, texturizarlos y animar los que sean necesarios. Se creará un ambiente lo más parecido posible al de la película mediante efectos de partículas, sonidos y luces de diferentes tipos, para así dotar a la escena de un ambiente tétrico y oscuro. Además, haré que sea compatible con gafas de realidad virtual y crear así un mayor efecto de inmersión.

Para la realización de esta escena 3D utilizaré el motor gráfico Unreal Engine 4(UE4) y otros programas externos para el modelado 3D y el texturizado.

Respecto a la programación, UE4 permite programar en #C++ o un sistema de scripting visual llamado “Blueprints”, que de manera medianamente sencilla te permite realizar la programación del juego sin necesidad de crear tu propio código; yo usaré el segundo.

Índice

Índice de figuras

1.Introducción

Hoy en día prácticamente todo está relacionado con el entretenimiento digital, el contenido multimedia como películas y videojuegos son el mejor ejemplo para esto. Tanto las películas como los videojuegos han alcanzado un apartado visual y unos efectos especiales impactantes que junto con el uso de la realidad virtual en los videojuegos crean experiencias inmersivas realmente alucinantes.

Actualmente, con la llegada de la realidad virtual, están surgiendo nuevas formas de jugar e interactuar con el usuario. Se crean experiencias que consiguen ponerte en la piel del personaje, como disfrutar de un viaje en montaña rusa, videojuegos completamente interactivos y jugables de principio a fin o infroarquitectura, con lo que se está cambiando la manera de enseñarle a un cliente de una manera hiperrealista el aspecto final de su futura vivienda.

Puesto que en la carrera se han abordado temas de modelado 3D y creación de videojuego, me ha parecido un tema interesante realizar un escenario 3D por el que poder movernos y con el que poder interactuar, más concretamente, crear el escenario de la película "*No respires"* en el motor gráfico Unreal Engine 4. En este documento se abordarán diferentes temas con una estructura muy definida para poder entender paso a paso todo el desarrollo del diseño de una escena jugable en 3D.

En primer lugar, hablaremos sobre el marco teórico, que es el estado actual de las tecnologías y el estado del arte hoy en día en el cual nos basaremos para crear la escena. A continuación, hablaremos de los objetivos que se han marcado, la metodología que se ha usado para poder cumplir con los objetivos y las herramientas utilizadas. Por último, hablaremos sobre todo del proceso del diseño 3D para videojuegos y la creación de entornos interactivos utilizando todos los elementos que hayamos diseñado.

2.Marco teórico o Estado del arte

En este bloque voy a explicar el concepto de cine y videojuego, mencionando el cambio que han sufrido desde sus inicios y como son actualmente, haciendo especial mención al género de terror, para así poder entender mejor el proyecto que voy a desarrollar y en qué producciones me he basado para la realización de este.

2.1 Cine

La definición de cine según la RAE es la siguiente: *"Película cinematográfica en la que se narra una historia real o imaginaria."*

El cine surgió como un espectáculo en París en el año 1985, a manos de los hermanos Lumière que crearon el cinematógrafo, el cual permitía tomar y proyectar imágenes generando el efecto de movimiento. Todo surgió con imágenes en blanco y negro y sin reproducir sonido. [1]

El cine fue evolucionando, mostrando imágenes en color y reproduciendo sonidos gracias a nuevos sistemas de reproducción hasta lo que conocemos hoy en día, con efectos especiales que permiten la creación total de escenarios, personajes y todo tipo de efectos ofreciendo un acabado visual impactante, permitiendo crear historias de todo tipo de manera realista.

El género de las reproducciones cinematográficas se divide en diferentes temáticas dependiendo del sentimiento que quieran producir en el espectador como, por ejemplo, drama, acción, romance o terror. Yo me voy a centrar en esta última, ya que voy a basarme en una de ellas para desarrollar el proyecto.

2.1.2 Cine de terror

La Wikipedia define el **cine de terror** como un género cinematográfico que se caracteriza por su voluntad de provocar en el espectador sensaciones de pavor, terror, miedo, disgusto, repugnancia, horror, incomodidad o preocupación.

En los inicios del cine ya se produjo la primera película de terror *L'arrivée d'un train à La Ciotat* en el año 1986. Fue algo tan simple como proyectar un tren que se acercaba cada vez más hacia los espectadores, pero que al ser este tipo de entretenimiento tan innovador y los espectadores no conocerlo, salían corriendo de la sala atemorizados por que creían que el tren iba a rollarlos.

Hoy en día gracias a los efectos especiales y la buena caracterización de los personajes se consiguen películas realmente terroríficas, y si a eso le añades el perfeccionamiento de las técnicas de grabación como los propios planos y los efectos sonoros, consiguen que realmente te sumerjas dentro de la película y que te asustes.

Existen varios subgéneros como por ejemplo el cine de *Monstruos*, donde una criatura siembra el pánico durante toda la película; *Slashers*, donde un psicópata se encarga de matar uno a uno a todos los protagonistas de manera muy gore, o *Paranormal*, que son aquellas en las que ocurren cosas científicamente imposibles.

A continuación, voy a comentar algunos ejemplos de películas de este género que estudié para realizar el proyecto.

2.1.2.1 Alien: El octavo pasajero

La nave de carga Nostramo, durante su regreso, interrumpe su viaje y despierta a sus siete tripulantes en estado de criogenización cuando el ordenador central, MADRE, detecta una transmisión procedente de un planeta cercano de origen desconocido. Cuando se acercan al planeta para ver qué sucede, se encuentran con una nave alienígena y mensajes de que dejen el planeta. De regreso a la nave son atacados por un organismo vivo, e ignorando el estado de cuarentena deciden meterse a la nave. Una vez a salvo se dan cuenta de que no están solos en la Nostramo y se desata el caos, muriendo uno a uno cada tripulante de la nave a manos de un ser alienígena muy evolucionado.

Ilustración 1. Escena de la película Alien: el octavo pasajero

Esta fue mi primera opción, quise imitar la nave donde se desarrolla la acción, pero era muy extensa y con escenarios muy similares y repetitivos a lo largo de toda la nave, por lo que lo descarté.

2.1.2.2 El resplandor

Esta es una de las películas más míticas del género de terror donde un padre junto a su familia se traslada a un hotel durante la época hibernal para el mantenimiento del hotel. Su intención es estar en un lugar tranquilo donde relajarse y terminar una novela que está escribiendo, pero pronto el padre va perdiendo la cabeza a la vez que pasan cosas paranormales en el hotel.

Ilustración 2. Esta es la escena más popular de la película el resplandor

La película se desarrolla en un hotel, pero no tiene un ambiente tétrico, que es lo que buscaba para desarrollar el proyecto, la película genera la sensación de terror por los sucesos que acontecen y no tanto por la ambientación.

2.1.2.3 Múltiple

Kevin posee 23 personalidades diferentes, pero queda una última y oscura personalidad por surgir. Kevin se ve obligado, por sus diferentes personalidades, a secuestrar a tres chicas jóvenes para poder llegar a ser, según ellas, un ser superior. Mientras tanto, Kevin lucha en su interior para poder controlar al resto de sus personalidades y evitar hacer daño a la gente que le rodea.

Ilustración 3 Cartel de la película Múltiple donde aparecen algunas de las personalidades del protagonista.

Esta fue también una de las primeras opciones que contemplé para realizar el proyecto, la guarida donde se esconde el protagonista es una serie de pasillos y habitaciones donde encierra a las víctimas, y tiene cierto ambiente tétrico y detalles con lo que podía hacer muchas referencias a la película.

2.1.2.4 No respires

Ésta es la película por la que me decanté para hacer el proyecto y realizar el entorno virtual.

El argumento de la película es que unos jóvenes ven la oportunidad perfecta para conseguir dinero fácil. Su objetivo es un hombre ciego que vive solo y que posee mucho dinero escondido en su casa, pero una vez dentro tendrán que luchar por sobrevivir y escapar de la casa al resultar que el ciego es un psicópata que esconde un gran secreto.

Ilustración 4. Escena de la película donde aparecen los 3 ladrones cuando se meten a robar a casa del ciego.

La película se desarrolla en una casa vieja con varias plantas diferenciadas y con un ambiente muy oscuro, por lo que decidí elegir esta película para desarrollar mi proyecto e intentar simular esa ambientación de terror y suspense que ofrece la película.

2.2 ¿Qué es un videojuego?

Para entender el proyecto que voy a hacer es necesario explicar también el concepto de videojuego. Según la Wikipedia:

"Un videojuego es un juego electrónico en el que una o más personas interactúan, por medio de un controlador, con un dispositivo que muestra imágenes de video".

Un videojuego es una de las mayores formas de entretenimiento de hoy en día y se pueden jugar en todo tipo de dispositivos electrónicos desde ordenadores y consolas de sobremesa, pasando por consolas portátiles, hasta móviles.

Para realmente ser consciente del alcance a la población con esta forma de entretenimiento, en 2016 se hizo un estudio sobe la población española donde se estimó que la gente que juega a videojuegos son 14,7 millones de jugadores (41% de la población) y donde las ventas del sector de videojuegos superaron los 1.000 millones de euros generados.

Los videojuegos empezaron con máquinas arcades representando unos pocos pixeles por pantalla, pero hoy en día han llegado casi al punto de parecerse a películas, con gráficos y efectos que imitan bastante bien la realidad, además, hoy en día se está desarrollando nuevas formas de interactuar con el juego, mediante tecnología de detección de movimiento o realidad virtual, como las HTC VIVE, para así aumentar la experiencia de inmersión del videojuego.

Ilustración 5. Imagen promocional de persona jugando a las gafas de realidad virtual HTC vive

2.3 Referencias

Una vez decidida la película que quería representar en UE4, busqué diferentes videojuegos de los últimos años que tuvieran una apariencia similar a la casa donde se desarrolla la acción, ambientación o mecánicas que pudiesen servirme como referencia y los estudié.

2.3.1 Outlast

Outlast es un videojuego de survival horror en primera persona desarrollado y publicado por Red Barrels Games el 4 de septiembre de 2014.

El protagonista es capaz de desplazarse, escalar u ocultarse en lugares de su entorno para así poder eludir a los enemigos ya que no podrá enfrentarse a ellos. La resistencia del personaje

es limitada y se puede recuperar progresivamente de las heridas si permanece un determinado tiempo quieto, sin ser atacado.

Dado que la mayoría del manicomio es oscuro, se debe usar una cámara para ver en la oscuridad gracias a la función de la visión nocturna. Sin embargo, el uso excesivo de la visión nocturna consumirá la batería y se necesitarán buscar más por los escenarios. [2]

La mayor parte de los enemigos en el juego son pacientes del manicomio que presentan mutaciones y comportamientos agresivos.

Ilustración 6. Captura de una cinemática del juego Outlast.

2.3.2 Resident Evil 7

Es un videojuego perteneciente al género del survival horror, desarrollado por la empresa japonesa Capcom, lanzado el 24 de Enero de 2017 para PC, Xbox one y PS4, esta última con soporte para PlayStation VR, lo que te permitía sentir una experiencia inmersiva realmente terrorífica.

Resident Evil 7 es el primer título de la saga principal que emplea una perspectiva en primera persona. A diferencia de otros videojuegos más cercanos al género del horror puro, como *Amnesia: The Dark Descent* y *Outlast*, es capaz de usar un considerable repertorio de armas para luchar contra los distintos enemigos. Varios tramos del juego consisten en ser

acechados por los miembros de la familia Baker, nuestros captores, quienes no pueden ser aniquilados en los primeros combates, solo pueden ser temporalmente incapacitados. Sin embargo, estos encuentros son evitables mediante el sigilo o huyendo.

El juego, como va siendo característicos en este tipo de géneros posee rompecabezas, la gestión de recursos en baúles de almacenamiento y combinación de distintos recursos para obtener consumibles que te den ventajas en ciertas ocasiones. [3]

El modo de juego enfatiza más el horror y la exploración sobre la acción.

Ilustración 7. Captura de una cinemática de uno de los momentos más conocidos de RE7.

Este juego tiene una ambientación tétrica y terrorífica, que junto a una iluminación tenue y efectos sonoros te harán pasarlo realmente mal mientras recorres la mansión derruida de la familia Baker.

2.3.3 P.T.

P.T. tiene como protagonista a un desconocido despertándose en una casa misteriosa, que conforme avanzas te vas dando cuenta de que está embrujada e irás explorando la casa solo con la ayuda de una linterna que te encontrarás al principio del juego.

Las únicas acciones que puede realizar el jugador es andar y hacer zoom. Para progresar, el jugador debe investigar eventos y resolver puzles que conforme los vas resolviendo, aparecen nuevos cambios en ese mismo escenario. [4]

Este juego, como lo que voy a hacer yo, más que un juego, es una experiencia, puedes moverte por el escenario e interactuar con ciertos objetos, pero nada más, sería más bien como una demo técnica. Al terminar el juego, su desarrollador Kojima, dijo literalmente que lo que quería conseguir con este juego es que fuese capaz de hacerte *"cagar en los pantalones".*

Aun no siendo un juego como tal, esta demo tuvo una gran acogida, llegando a alcanzar un millón de descargas en PS4.

Ilustración 8. Imagen in-game de P.T.

3.Objetivos

3.1 Objetivos generales

El objetivo general del TFG será la simulación en Unreal Engine 4 del escenario que aparece en la película "*No respires"*, creando yo todos los *assets*[1] y texturizándolos lo más fielmente posible a la película, haciendo uso de programas de modelado 3D, escultura y texturizado que explicaré en profundidad más adelante. Dotaré la escena de un ambiente oscuro y de terror con la ayuda de sistemas de partículas y diferentes tipos de iluminación e introduciré guiños que entenderán aquellos que hayan visto la película.

Además, se contemplará la posibilidad de introducir el uso de gafas de realidad aumentada para mejorar la experiencia.

3.2 Desglose de objetivos específicos.

1. Crear los planos de la casa de la película *"No respires"* y generarla en 3D.
2. Creación de todos y cada uno de los elementos 3D de manera realista tal y como aparecen en la película con los programas 3ds Max y MudBox.
3. Texturizado con Substance Painter, Photoshop y CrazyBump de todos los objetos que haya creado.
4. Importar los objetos creados adaptándolos al motor gráfico UE4.
5. Crear una iluminación realista y efectos de partículas para dar una buena ambientación de terror a la escena en UE4.
6. Crear mediante el sistema de scripting de BluePrints que ofrece Unreal Engine las mecánicas del juego.
7. Crear un escenario en Unreal Engine que te permita explorar las diferentes habitaciones de la casa y con el que puedas interactuar con ciertos objetos.
8. Crear un ejecutable para poder jugar desde cualquier ordenador al juego.
9. Hacer compatible el videojuego con el dispositivo de realidad virtual Oculus Rift.

[1] Assets: Son cada uno de los elementos que componen el juego (animaciones, modelos, IA, sonidos, etc.).

4. Metodología

4.1 Metodología de desarrollo

Para llevar a cabo la gestión del proyecto y llevar un control sobre las tareas que tenía que realizar y cumplir los plazos que me marcaba de la mejor manera posible, utilicé la metodología de desarrollo Kanban.

4.1.1 Kanban

En el desarrollo de software, se utiliza el sistema Kanban virtual para limitar el trabajo en curso. Se utilizan tarjetas que representan los elementos de trabajo o tareas a realizar.

El método Kanban formulado por David J. Anderson es una aproximación al proceso gradual, evolutivo y al cambio de sistemas para las organizaciones.

Prácticas del método Kanban:

1. **Visualizar**

   Visualizar el flujo de trabajo y hacerlo visible es la base para comprender cómo avanza el trabajo. Sin comprender el flujo de trabajo, realizar los cambios adecuados es más difícil. Una forma común de visualizar el flujo de trabajo es el uso de columnas. Las columnas representan los diferentes estados o pasos en el flujo de trabajo.

2. **Limitar el trabajo en curso**

3. **Dirigir y gestionar el flujo**

   Al gestionar activamente el flujo, los cambios continuos, graduales y evolutivos del sistema pueden ser evaluados para tener efectos positivos o negativos.

4. **Hacer las Políticas de Proceso Explícitas**

   Las políticas definirán cuándo y por qué una tarjeta debe pasar de una columna a otra.

5. **Utilizar modelos para reconocer oportunidades de mejora**

   Cuando los equipos tienen un entendimiento común de las teorías sobre el trabajo, el flujo de trabajo, el proceso y el riesgo, es más probable que sea capaz de construir

una comprensión compartida de un problema y proponer acciones de mejora que puedan ser aprobadas por consenso. [5]

¿Por qué elegí Kanban?

A pesar de que voy a realizar un proyecto individual, Kanban me permite llevar un control en todo momento de las tareas que tengo que hacer y las que me quedan.

Puesto que me limita el flujo de trabajo mediante una cola, hasta que no termine con una tarea no empezaría con la siguiente, y así, podría conseguir una evolución continua e incremental con la que identificar problemas y a la vez evitarlos.

Si empezamos por ejemplo con una base sólida, y hasta que no esté completamente acabada no seguimos con elementos que dependen en parte de esa base, nos ahorraríamos el problema de que a posteriori tuviésemos que hacer un cambio en la base que provocaría también cambiar el resto de las tareas que pudiésemos tener ya acabadas.

Gracias al método Kanban, nos podemos marcar objetivos para tener acabados semanalmente y llevar un control sobre los tiempos de entrega.

4.2 Gestión del proyecto con Trello

La gestión del proyecto la he llevado a cabo mediante un tablón Kanban con la herramienta Trello.

Trello es una página web que te permite la creación de tablones para realizar la metodología Kanban. Básicamente, lo que te permite Trello es, dividir por columnas según el estado de una tarjeta (por hacer, en proceso y finalizado), e ir cambiando el estado de estas conforme se vayan completando. Las tarjetas contienen las tareas a realizar o subtareas, además, puedes añadir los porcentajes de estado de la tarea, comentarios y asignar tareas a los diferentes integrantes del equipo de desarrollo. [6]

Por último, también puedes asignarles a las tarjetas colores para identificar qué tipo de tarea es.

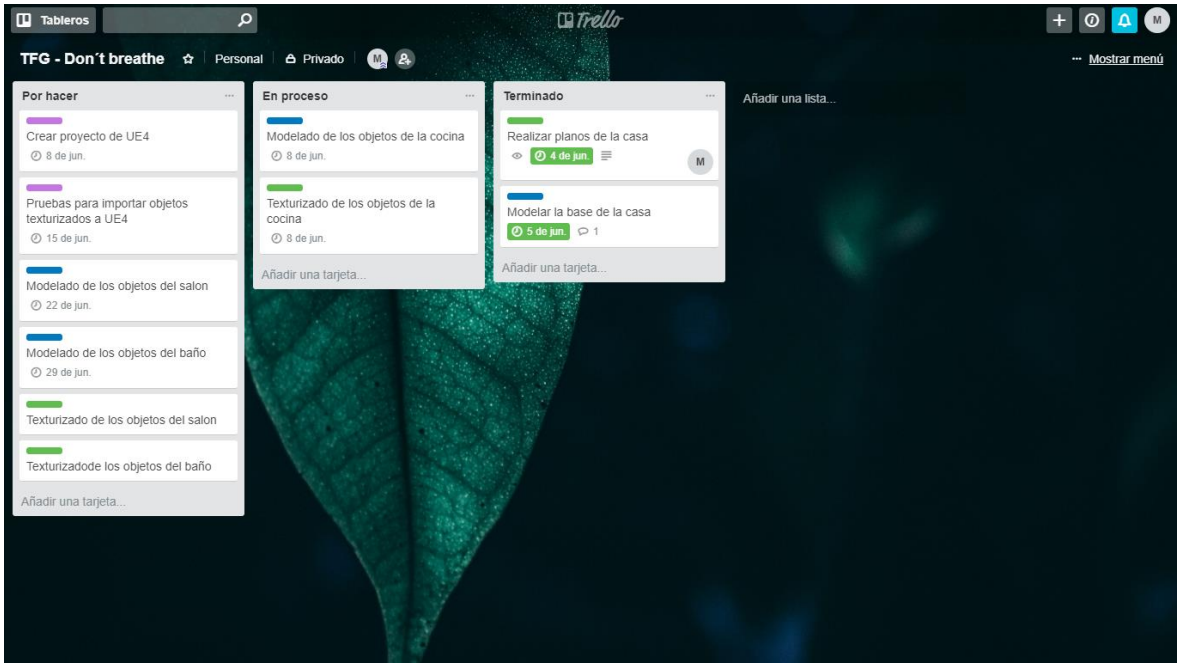


Ilustración 9. Tablón de Trello.

4.3 Toggle

Toggle es una web que te permite gestionar proyectos, creando tareas que se pueden categorizar mediante etiquetas y medirte los tiempos para tener un control del tiempo dedicado a estas y así no perder tiempos excesivos y poder así tener las tareas programadas al día.

También te permite tener una visión general de las horas dedicadas al proyecto, filtrando por fechas y por categorías.

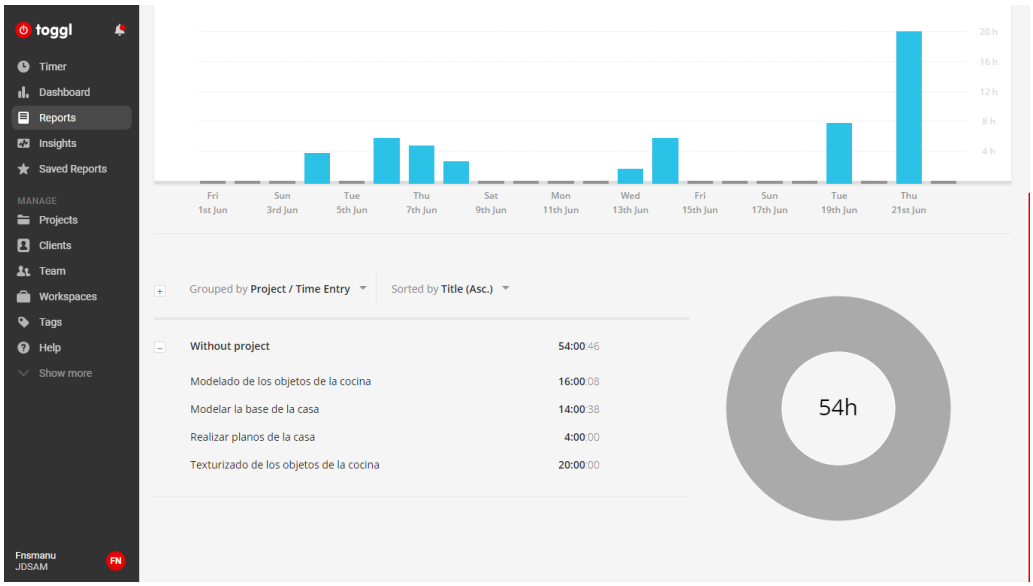


Ilustración 10. Imagen de un mes de ejemplo de cómo sería el control del tiempo.

4.4 Control de versiones

Control de versiones como tal no he utilizado, los repositorios tipo GitHub, no los he visto de gran utilidad para la clase de proyecto que estaba realizando. Es un proyecto individual y los archivos eran bastante pesados, de varios GB, y decidí no llevar un control de versiones como tal. Pero sí me he percatado de tener siempre varias copias por si surgía algún problema, en Dropbox siempre tenía 3 carpetas con diferentes versiones, guardando las últimas 3 versiones funcionales que tenía para poder volver a alguna de ellas si surgía cualquier tipo de problema.

5. Herramientas que componen el Workflow

5.1 Motor gráfico

Para llevar a cabo la elaboración del proyecto que quiero realizar necesito crear un entorno 3D por el que moverme e interactuar con los diferentes objetos, aquí es donde entra en juego el motor gráfico.

5.1.1 ¿Qué es?

Un motor gráfico es un framework de software diseñado para crear **y** desarrollar videojuegos. Los desarrolladores de videojuegos pueden usar los motores para crear videojuegos para una consola, dispositivo móvil o un ordenador.

Todo motor gráfico ha de ofrecer al programador unas funcionalidades básicas, proporcionando normalmente un motor de renderizado para gráficos 2D y 3D, un motor que detecte la colisión física de objetos y la respuesta a dicha colisión, sonidos y música, animación, inteligencia artificial, comunicación con la red para juegos multijugador, posibilidad de ejecución en hilos, gestión de memoria o soporte para localización. [7]

5.1.2 Comparativa mejores motores gráficos del mercado

Hoy en día existen motores gráficos muy potentes capaces de crear imágenes hiperrealistas de una manera medianamente sencilla en comparación a como se hacía antes y muy fluida. En el mercado hay muchos motores gráficos que podemos usar, ya sean de manera gratuita o con algún tipo de licencia. A continuación, voy nombrar los mejores motores del mercado que existen actualmente.

5.1.2.1 Unity 5

La nueva versión incluye una “mejora inmensa en las capacidades gráficas”, como la iluminación en tiempo real, shader de base física construidos de materiales del mundo real

y reflejos HDR. Algunos juegos populares que se han creado son *Deus EX: The Fall, Heartstone, Rust* y *Assassin's Creed Identity*

Unity es un motor gráfico históricamente asociado a los juegos para dispositivos móviles, pero el lanzamiento de Unity 5 y su nuevo sistema de renderizado suponen nuevas y potentes capacidades para crear juegos con gráficos realistas.

Una de las claves de Unity es que el juego puede ser portado a 25 plataformas diferentes empleando un solo código**,** lo cual facilita crear una versión de su juego para varias plataformas, y la documentación y comunidad relacionado con este motor es muy extensa.

Dependiendo de para qué quieras el motor gráfico el tipo de licencia varía y el coste aumenta; para estudiantes y gente que quiera aprender sería completamente gratuito, pero si quieres Unity para crear un videojuego profesional y comercializarlo, tendrás que pagar la licencia. Existen dos versiones de pago, Plus y Pro, que cuestan 25$/mes y 125$/mes respectivamente.

5.1.2.2 CryEngine

CryEngine es un motor gráfico extremadamente potente, desarrollado por Crytek e introducido con su primer *Far Cry*. El motor está diseñado para usarse en juegos de PC y consolas, incluyendo PlayStation 4 y Xbox One.

CryEngine V es la última versión de este motor y aseguran que es la solución más accesible, potente y repleta de características que hayan creado hasta el momento. A su vez ofrece nuevas opciones: una nueva API para los desarrolladores de C# para trabajar con scripts de inmediato; un incremento significativo del rendimiento de hardware actual; compatibilidad

con DirectX 12**;** creación de asombrosos efectos de fluidos en tiempo real y una interfaz más intuitiva que la anterior, ya que tenía una curva de aprendizaje muy elevada; una mayor flexibilidad en los efectos sonoros; por último, un canal dedicado a la comunidad para compartir opiniones.

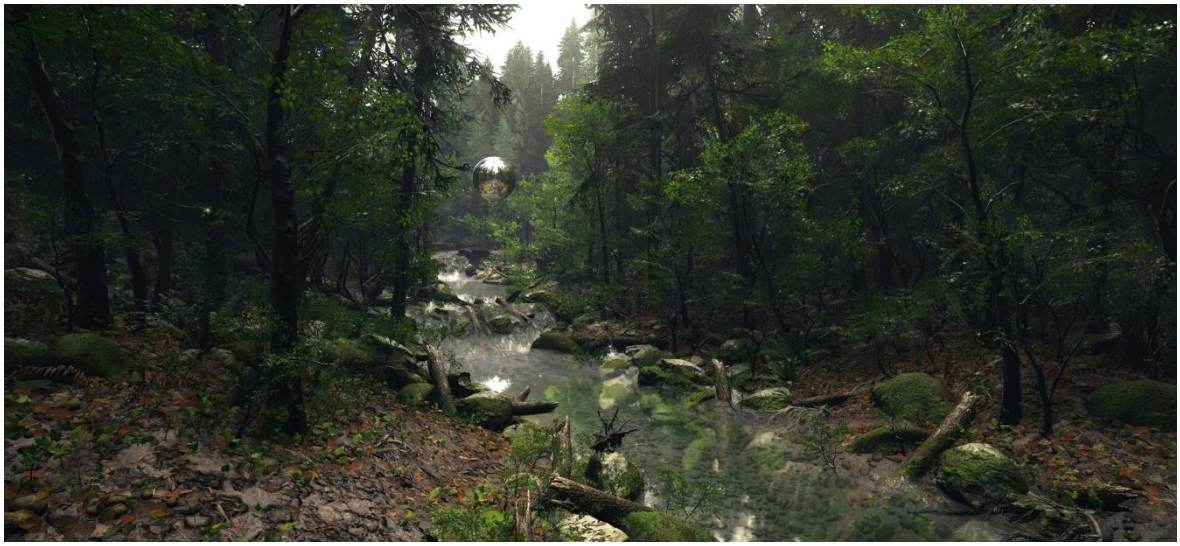


Ilustración 11. Escena en Cry Engine

Además, desde Crytek quieren apostar fuerte por la realidad virtual (VR) y prometen que CryEngine V será el mejor soporte para crear experiencias inolvidables en PlayStation VR, OSVR, HTC Vive y Oculus Rift.

Para que todos los desarrolladores puedan disponer de él los creativos tendrán un completo acceso al set de características de este motor y a todo su código por la cantidad de dinero que ellos mismos consideren.

5.1.2.3 Unreal Engine 4

Unreal Engine 4 (UE4) es el nuevo motor gráfico de Epic Games, y sucesor del Unreal Development Kit. Tiene unas capacidades gráficas muy potentes, incluida la iluminación dinámica y un sistema de partículas que permite manejar un millón de partículas en una misma escena.

Con Unreal Engine 4 se han creado juegos tan populares como la saga *Gears of War*, *Unreal Tournament*, *Fortnite* y *Players Unknown Battleground.*

Ilustración 12. Escena en Unreal Engine 4

La curva de aprendizaje de este motor gráfico es bastante baja, puedes crear tus primeras escenas de manera sencilla y un sistema de creación de materiales bastante intuitivo mediante nodos, y el tema de programación se puede hacer en C# o con un sistema de scripting visual bastante sencillo llamado Blueprints.

También se puede utilizar sistemas de realidad virtual como OculustRift y HTC vive.

Unreal Engine 4 es totalmente gratuito en la actualidad, aunque tendrás que pagar un 5% en royalties a no ser que ganes menos de $3.000 por trimestre y juego. Es una de las opciones ideales para empezar a desarrollar juegos o si tu estudio no es muy grande.

5.1.3 Elección del motor

Un buen motor gráfico es el que traslada tus ideas creativas fácilmente a gráficos en una pantalla**.** Por ello, mi elección ha sido Unreal Engine 4, ya que, para realizar un proyecto inicial, puedes usar una plantilla dependiendo del tipo de juego que quieras hacer, y a partir de ahí ir añadiéndole los modelados 3D y texturas. Además, UE4 destaca por el buen acabado en el diseño de interiores, y puesto que es lo que voy a hacer viene perfecto.

Por otro lado, el sistema de programación mediante Blueprints es más intuitivo y sencillo que programar en C#, por lo que las funcionalidades adicionales que me hagan falta las añadiré con este sistema de scripting.

Aunque Unity 5 tiene una comunidad y documentación difícil de superar, Unreal Engine 4 tampoco está nada mal, con una comunidad activa e incluso desarrolladores de la propia Epic, que hacen videotutoriales que ayudan mucho.

Por último, lo que me ha terminado de convencer para la elección de este motor es que es completamente gratuito desarrollar el juego, ellos se quedarán un porcentaje de las ganancias si supera cierta cifra en ventas.

5.2 Modelado 3D

5.2.1 ¿Qué es?

El **modelado 3D** es el proceso de desarrollo de una representación matemática de cualquier objeto tridimensional a través de un software especializado. Se puede visualizar como una imagen bidimensional mediante un proceso llamado renderizado 3D o utilizar en una simulación. El modelo también se puede crear físicamente usando dispositivos de impresión 3D.

Los modelos 3D representan un objeto tridimensional usando una colección de puntos en el espacio dentro de un espacio 3D, conectados por varias entidades geométricas tales como triángulos, líneas, superficies curvas, etc. Siendo una colección de datos (puntos y otro tipo de información), los modelos 3D pueden ser hechos a mano, a través de algoritmos o bien escaneados.

Existen diferentes tipos de modelado:

Modelado poligonal - Son puntos en un espacio 3D, llamados vértices, que están conectados para formar una malla poligonal. Los polígonos son planos y solamente se pueden aproximar a superficies curvas usando varios polígonos.

Modelado de curvas -Las superficies están definidas por curvas, las cuales son influenciadas por la ponderación del control de puntos.

Escultura digital – En este tipo de modelado se trabaja con una cantidad muy alta de polígonos que se esculpen deformando la superficie. Puesto que suelen tener un peso tan grande, se suelen utilizar para crear los mapas de normales (que explicaremos más adelante) para posteriormente aplicárselo a un modelo poligonal con muchos menos polígonos. [8]

5.2.2 Herramientas

Para la realización de todos los modelados 3D que voy a realizar usaré, dependiendo de mi objetivo, uno de los siguientes programas o ambos, ya que se pueden complementar.

5.2.2.1 Autodesk 3ds Max

3ds Max es un Software de renderización, animación y modelado en 3D desarrollado por Autodesk.

3ds Max, con su arquitectura basada en plugins, es uno de los programas de animación 3D más utilizado, especialmente para la creación de videojuegos, anuncios de televisión, en arquitectura o en películas.

Nosotros lo usaremos para la creación de los **modelados poligonales** y los mapas de texturas UV, con una licencia de 1 año de estudiante.

5.2.2.2 MudBox

Mudbox es un software de modelado 3D, texturizado y pintura digital, actualmente desarrollado por Autodesk. Mudbox fue creado por Skymatter y fue usado por primera vez en la película King Kong (película de 2005).

A diferencia de 3ds Max, no puedes crear los modelados 3D directamente en este programa, necesitas importar objetos previamente creados en programas de modelado como 3ds Max o Maya. Mudbox, se utiliza para esculpir y crear modelados orgánicos como personajes o esculturas, también permite crear las texturas de manera medianamente sencilla, pero yo solo lo voy a utilizar para **escultura digital**.

La licencia de este programa también es de pago, pero puedes conseguir una de manera gratuita por ser estudiante.

5.3 Materiales y Texturas

5.3.1 ¿Qué son?

La palabra **material** etimológicamente quiere decir “relativo a la materia” por lo tanto es una característica de los objetos. En el campo del modelado 3D, los materiales funcionan de una manera similar; son un recubrimiento que se le aplica al objeto 3D para así poder deducir la procedencia de este, es decir, darle un aspecto metálico, plástico, vidrio, etc.

Los materiales poseen diferentes características que, con la combinación y la intensidad correcta, dotarán a los objetos de un aspecto realista. Estas son las características principales que ya explicaré más en profundidad más adelante: brillo, luminosidad, color, textura, densidad, extensión.

La **textura** es una propiedad que se le da a un material. Es la forma en la que están entrelazadas las fibras de un material, lo que produce una sensación táctil cuando la vemos, ya que produce sombras y da esa sensación de relieve. [9]

Ilustración 13. Comparación de materiales simples (izquierda) y materiales con texturas (derecha).

5.3.2 Herramientas

Para realizar el texturizado de los objetos y obtener un resultado realista existen diferentes herramientas que nos ayudarán a conseguir resultados rápidos y de buena calidad. Cada una de estas herramientas tiene diferentes funcionalidades. Con solo alguna de ellas se podría hacer todo el proceso, pero con la combinación de todas ellas se puede conseguir un resultado mucho más profesional si se complementan bien.

5.3.2.1.1 CrazyBump

Este es un programa que nos permite crear Bump Mapping, es decir, el mapa de normales, a partir de una imagen y así conseguir un acabado realista de manera muy sencilla e intuitiva.

Además, permite cargar modelos 3D con extensión .obj y visualizar al instante el acabado de la textura que has creado.

CrazyBump necesita una licencia de pago, pero podemos adquirir una de prueba de 30 días de manera gratuita.

5.3.2.1.2 Adobe Photoshop

Photoshop es el nombre popular de este programa informático de edición de imágenes. Está desarrollado por la empresa Adobe Systems Incorporated y funciona en los sistemas operativos Apple Macintosh y Windows.

De un modo general, Photoshop permite modificar imágenes digitalizadas, especialmente fotografías. También se utiliza para crear y editar imágenes y gráficos. La forma, la luz, el color y el fondo son algunos de los aspectos que esta herramienta permite editar mediante el uso de capas y mascarás, permitiéndote hacer prácticamente cualquier tipo de retoque fotográfico y diseño.

5.3.2.1.2.3 Substance Painter

Substance Painter es una herramienta que nos permite pintar materiales completos en objetos 3D en tiempo real.

Se define un material o materiales procedurales a través de un conjunto de canales donde cada uno controla los parámetros utilizados para generar las texturas que describen los atributos de superficie específicos, como difusa, especular y normales.

Este software de pintura 3D tiene características que no se habían visto antes, mejorando el flujo de trabajo y la creación de texturas, está reconocido como la aplicación 3D más innovadora y fácil de usar que hay para texturizar.

Este es un programa de pago, pero tiene licencias gratuitas de 1 año para estudiantes. Si no eres estudiantes puedes adquirir la licencia por unos 150$.

6.Cuerpo del Trabajo

6.1 Diseño del GDD

El GDD (Game Design Document) es una síntesis de lo que va a ser el videojuego (concepto, historia, género, número de plataformas, equipo de producción…).

Como el proyecto no va a ser un videojuego en sentido estricto, solo utilizaremos algunos aspectos de los apartados del GDD para ayudarnos a desarrollar de manera correcta nuestro escenario y la interacción con el mismo. [10]

6.1.1 Argumento

El argumento del juego acontece después de la historia de la película en cuestión. La película trata sobre unos jóvenes que entran a robar a la casa de un ciego, pero una vez dentro se dan cuenta que el ciego esconde en su sótano a una joven secuestrada; mientras tanto el ciego intenta dar caza a los ladrones. Nuestro personaje es un curioso que se cuela dentro de la

casa donde han sucedido los hechos y va recorriendo todas las habitaciones. Durante el recorrido verá sangre, escuchará sonidos extraños y encontrará el sótano donde tenía a la joven secuestrada.

6.1.2 Características básicas

Aquí resumo de manera rápida las características básicas que tendrá el juego:

Género: Terror y suspense.

Público Objetivo: PEGI 16

Plataforma: PC, compatibilidad con Oculust Rift.

Idioma: español.

6.1.3 Mecánicas del juego

Esta sección esencialmente describe lo que el jugador puede hacer y cómo puede hacerlo.

Cámara: La cámara del jugador será en primera persona, y no se verá nada referente al personaje, ni piernas ni manos a la hora de interactuar.

Acciones: Las acciones que se podrán hacer son saltar, interactuar con puertas para abrirlas, moverte y mirar alrededor.

Controles: Los controles del juego para poder moverlos en la escena varían dependiendo del periférico, podemos usar ratón y teclado o el mando de Xbox One y las Oculus Rift.

Ilustración 14. Controles del juego por teclado

Ilustración 15. Controles para VR

6.1.4 Apariencia y ambientación

La apariencia del juego, como en todos los de su estilo, será un aspecto de terror y que consiga ponerte en tensión. Se conseguirá creando materiales con un aspecto sucio y deteriorado que, junto con la iluminación, que se hará solamente mediante luces tenues procedentes de lámparas, se creará una ambientación tenebrosa. A continuación, muestro una imagen de un videojuego en el que me he basado (*P.T*) donde se ve cómo utiliza estos aspectos, y también imágenes de la película *No respires*, la cual intentaremos recrear e imitar.

Ilustración 16. Imágenes de referencia para la ambientación del juego

6.1.5 Sonidos

Los sonidos del juego se pueden dividir en dos.

Música de fondo: de fondo sonará una música continua que tendrá sus altos y bajos que creará un efecto de tensión para que cree momentos en los que poder hacer que el jugador se sobresalte.

Sonidos puntuales: mediante eventos se activarán ciertos sonidos llegado a algún punto de la escena para dotar de vida a la casa (sonido de cuervo, crujidos de madera, luces led intermitentes, ramas pisándolas).

6.1.6 Detalles de producción

En este apartado se definen las fechas estimadas para cada etapa de producción.

Comienzo de la producción: 1 de octubre. En esta etapa se empezarán a desarrollar todos los modelados 3D y texturizado.

Integrar modelados con Unreal Engine: 20 de enero. Aquí se empezará a importar todos los *assets* creados a Unreal Engine y a crear las funcionalidades.

Primera Demo jugable: 1 de marzo. Primera demo jugable para detectar errores y solucionarlos.

Fin de la producción: 15 de julio. Render final y empaquetado del juego.

6.2 Recopilación de información de la película

Para comenzar a construir y diseñar la escena, puesto que no tenía los planos de la casa ni nada que me pudiese servir, lo primero que hice fue conseguir la película y verla una vez entera para coger la idea general. A continuación, la volví a ver otra vez parándola cada vez que aparecía una escena con objetos de interés o parte de la distribución de la casa para poder hacer un plano, y así me fui apuntando en una lista el minuto en el que aparecía junto con una descripción, para poder volver ahí cuando quisiera y revisar de nuevo los detalles de la escena cuando me hiciera falta. Esto me llevó mucho tiempo; saber la distribución y proporciones de la casa de manera más o menos fiel fue una tarea complicada.

Casa general - 00:11:14	Armario caja fuerte - 00:20:26 // 00:34:00
Patio - 00:15:00	Recibidor - 00:21:45
Cocina - 00:20:00 // 00:31:50	Habitación ciego - 00:22:31
Baño - 00:32:20	Habitación Hija – 00:54:20
Taller - 00:20:35 // 00:59:00	Sótano – 00:39:40
Comedor - 00:20:55 // 00:25:00 // 00:38:00	Chica secuestrada – 00:42:20 // 1:08:00

Después de ver varias veces la película conseguí hacer un esquema general de la casa y de la distribución de todas las habitaciones y dibujé un plano en Photoshop que luego levantaría en 3D.

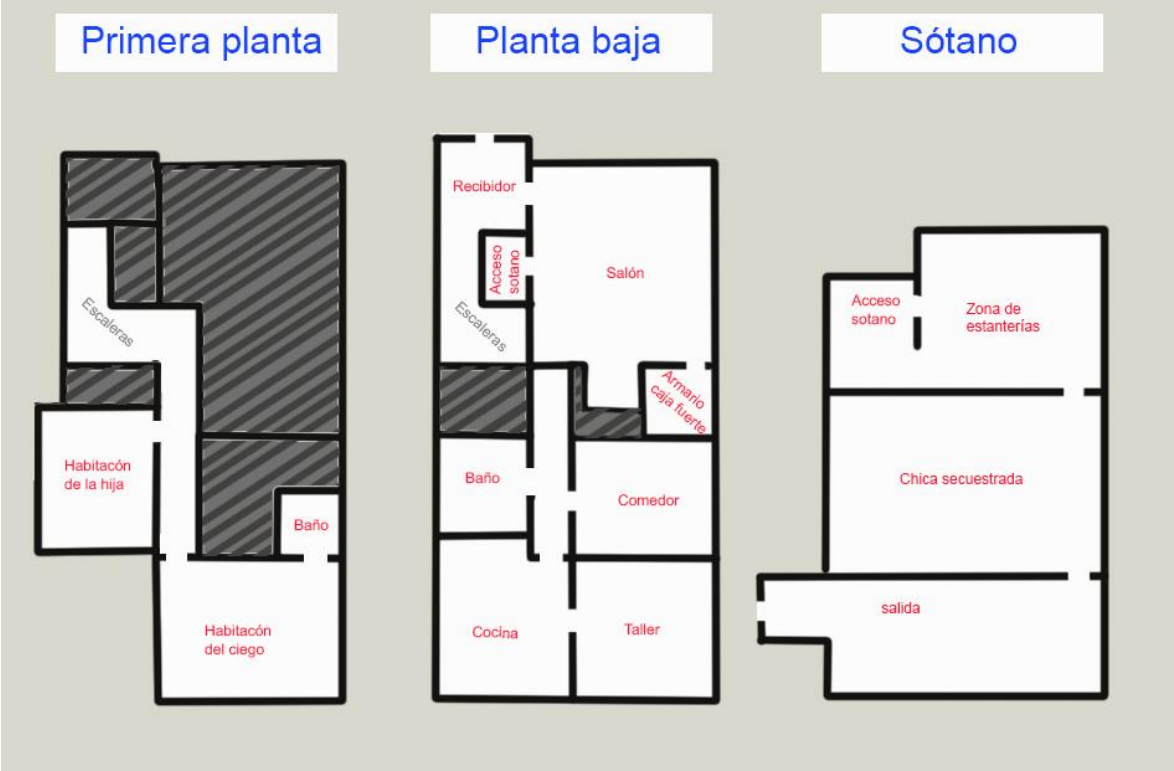


Ilustración 17.Planos de la casa de la película

6.3 Flujo de trabajo

En los siguientes puntos se explica detalladamente cómo se realizan los modelados 3D paso a paso, la creación de las UV y cómo se texturizan eligiendo los materiales que mejor convengan y por qué.

Por último, se explicará cómo se importan los *assets* a Unreal Engine 4.

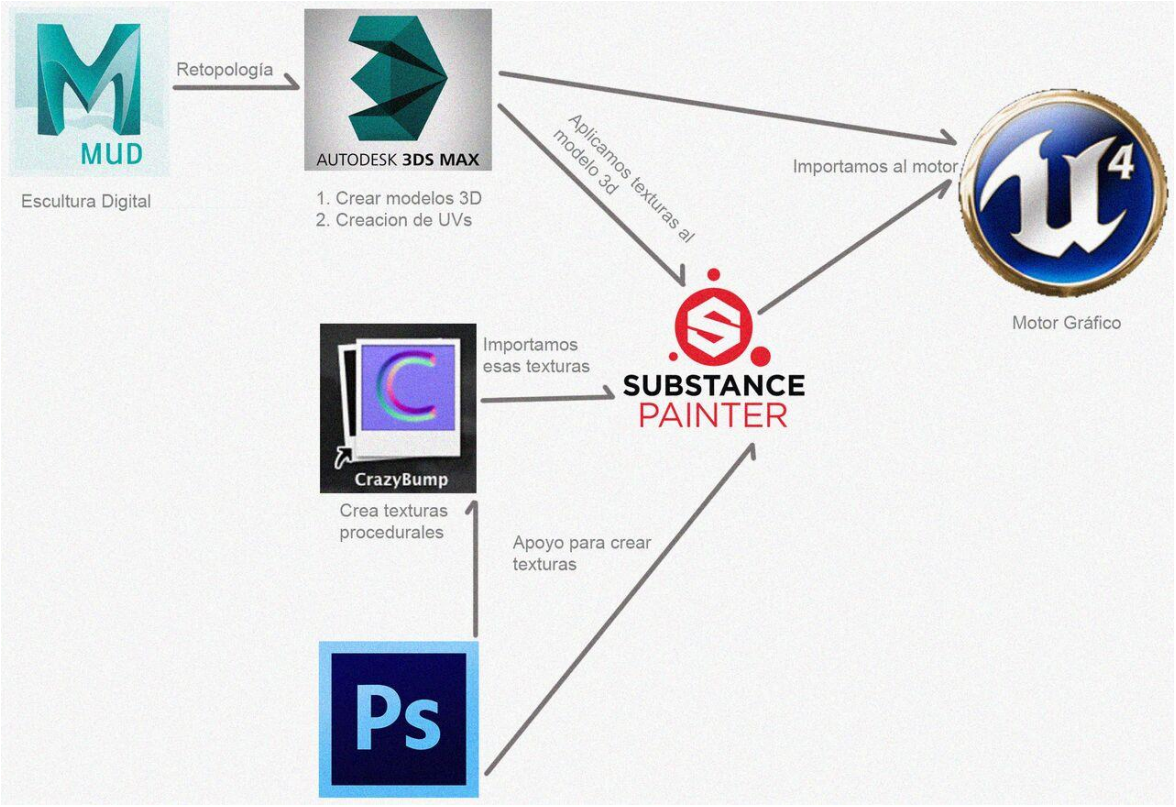


Ilustración 18. Flujo de trabajo

6.3.1 Diseño del arte

En este apartado explico cómo he llevado a cabo los diferentes apartados del diseño de los objetos y los procesos que lo involucran. Los procesos a nivel general son: modelado 3D, mapeado de texturas, creación de texturas.

6.3.1.1 Modelado de objetos

En este apartado voy a explicar cómo he hecho algunos de los modelados y los mecanismos para llevarlos a cabo, la importancia de la cantidad de polígonos y la topología[2] que debe tener el objeto.

[2] La topología es la forma en la que se distribuirán los polígonos a la hora de realizar un objetos 3D para su correcto funcionamiento para procesos futuros.

6.3.1.1.1 Modelado poligonal

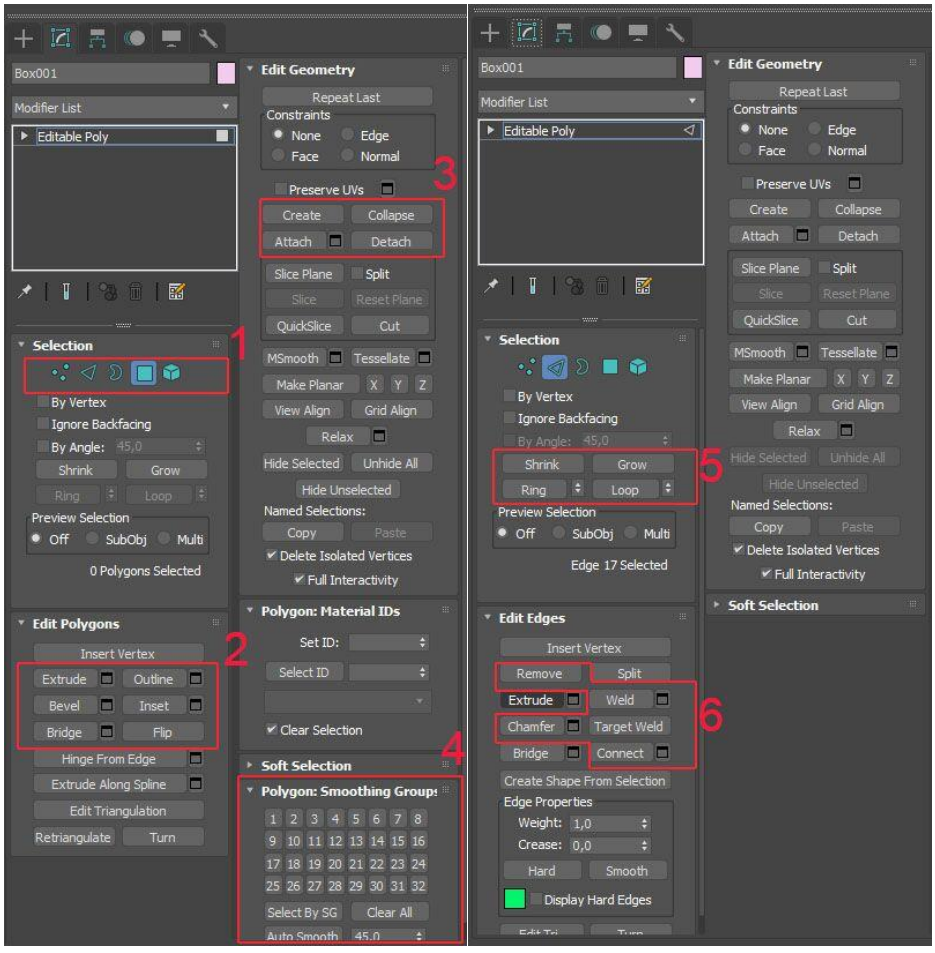
Este es el tipo de modelado predominante que voy a realizar. Consiste en realizar un modelo Low Poly que nos permitirá dotar a toda la escena de muchos de los objetos que aparecen en la película sin sobrecargar la escena y así tener la mejor optimización posible.

Para entender cómo se hacen estos modelados voy a realizar un tutorial que incluya la mayoría de las técnicas y modificadores que suelo utilizar en cada modelo.

Técnicas y modificadores más utilizadas.

Dentro del modo "Editable Poly" encontramos diferentes formas de editar la geometría y algunos modificadores con funciones muy específicas. A continuación, explicaré las que suelo utilizar siempre en cualquier modelado y profundizaré más adelante en algún modificador más específico para los diferentes ejemplos.

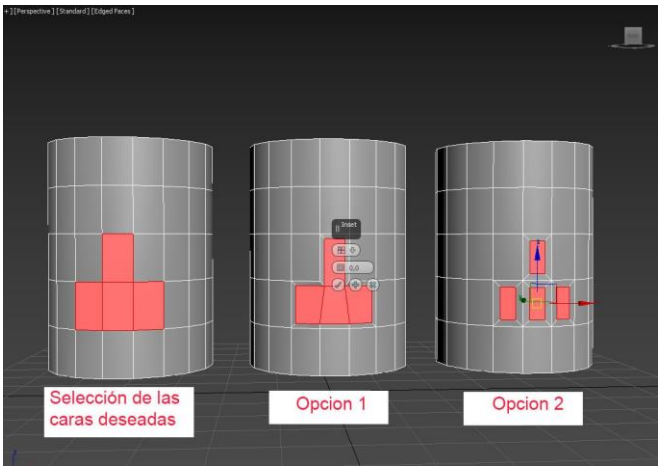
En esta captura vemos las opciones más básicas para editar la geometría del modelo 3D, explicaré las que tienen el recuadro rojo y pondré algún ejemplo visual.



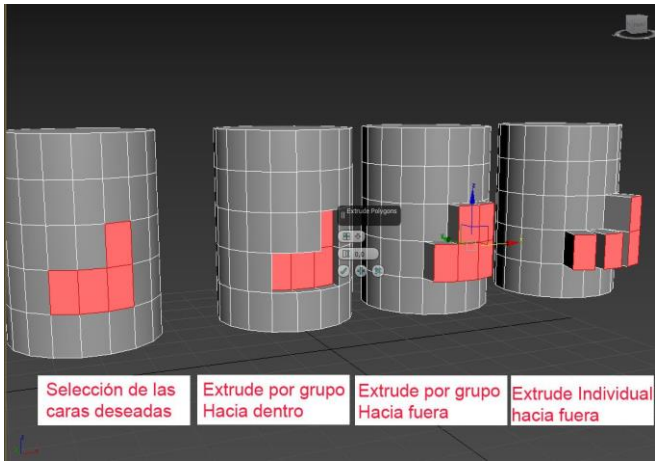
1) Esto te permite seleccionar las diferentes partes de la malla según vértices, lados, o polígonos. En la imagen se ve la selección por polígonos, dependiendo de qué tipo de selección tengamos; variarán un poco el resto de las opciones del menú. Una vez tengamos algo seleccionado, podemos moverlo, rotarlo o escalarlo, como opciones básicas.

2) Estas opciones aparecen al tener la opción de selección por polígonos, voy a explicar solo 3 de ellas ya que el resto que aparecen derivan de las otras:

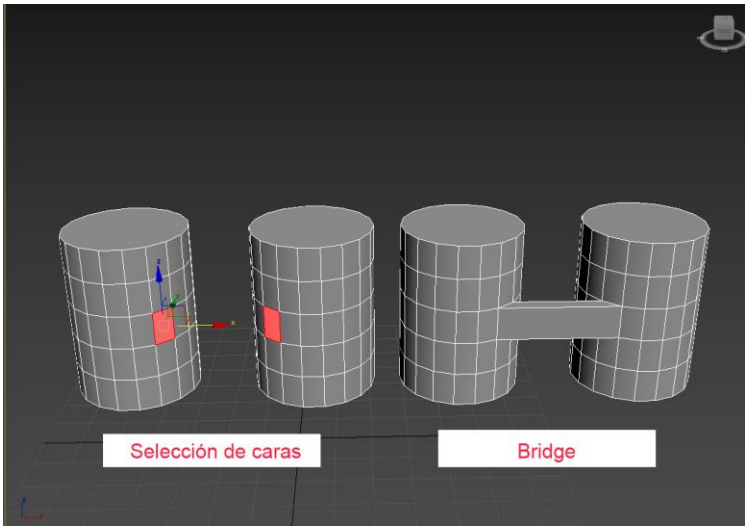
- **Inset:** Cuando seleccionas las caras y haces inset, crea nuevas líneas en el interior con la misma forma que las exteriores. Si haces una selección de varias caras, puedes hacer inset en cada cara seleccionada o un inset general de todas las caras seleccionadas mediante una opción que aparece en pantalla, que además te permite ajustar la distancia a la que se crearán las nuevas líneas.



- **Extrude:** cuando seleccionas una o varias caras, te permite extruirlas hacia dentro o hacia afuera. Permite extruir el conjunto de las caras o cada cara por separada, y la distancia que quieres extruir.



- **Bridge:** permite unir dos caras seleccionadas mediante un puente.



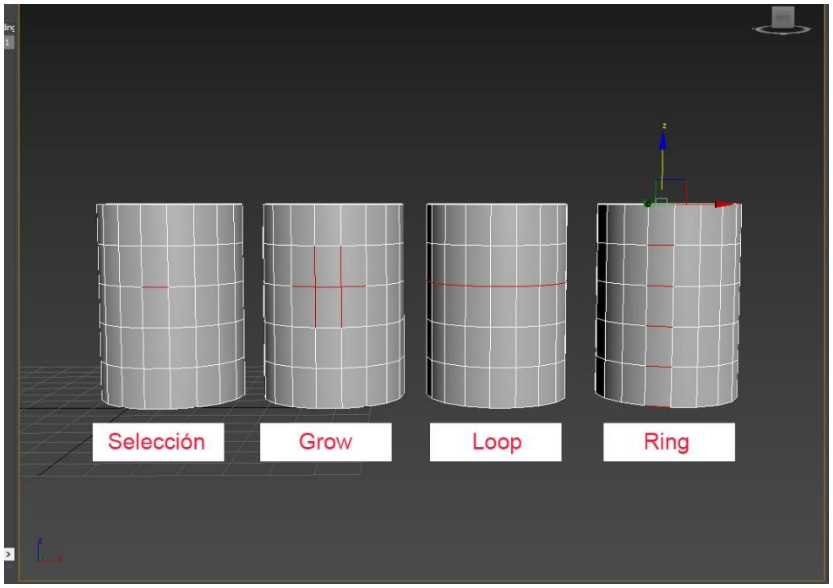
3) Este nos permite editar la geometría

- **Create:** crea un conjunto de vértices que al cerrarlos crean un polígono.
- **Collapse:** junta dos partes seleccionadas y la hace uno, es más útil cuando lo utilizas cuando estas en la selección por vértices, ya que te permite de manera rápida juntar dos vértices que estén separados.
- **Attach/Detach:** permite convertir varios objetos de la escena que estaban separados en un solo objeto, o por el contrario separarlos en varios lo que antes era uno.

4) Smoothing groups te permite aplicar a un conjunto de polígonos un suavizado para que no se note tanto el cambio entre polígonos, permitiéndote crear diferentes suavizados para diferentes partes del objeto.

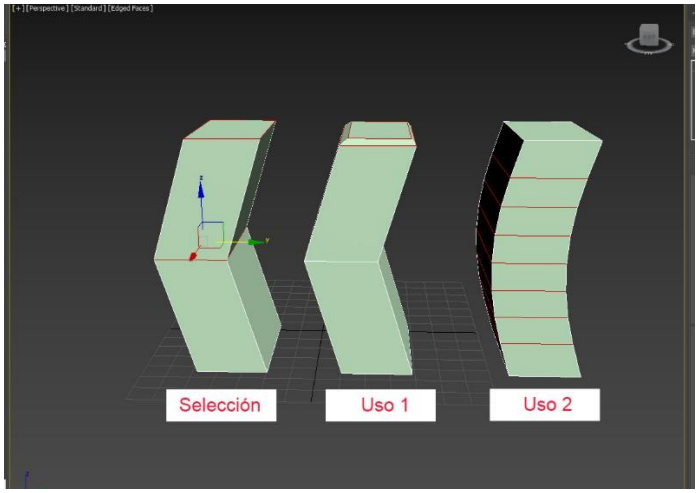
5) Nos permite seleccionar de manera rápida la cantidad de polígonos según su distribución.

- **Grow:** aumenta la cantidad de bordes seleccionados que están colindantes a él en todas direcciones.
- **Ring:** selecciona todos los bordes paralelos a él formando una especie de anillo.
- **Loop:** selecciona todos los bordes colindantes a él por el vértice, formando una línea.



6)Esta sección aparece en edición por bordes y vértices.

- **Remove:** esta es la manera correcta para borrar bordes y vértices.
- **Weld:** se utiliza para unir normalmente vértices, puedes seleccionar un grupo de vértices y establecer una distancia. Todos los vértices que estén por debajo de esa distancia unos de otros quedarán unidos.
- **Connect**: seleccionando varios bordes te permite crear nuevos, que se conectan entre sí. Permite añadir parámetros para el desplazamiento o separación de los nuevos bordes creados.
- **Chamfer**: crea un chaflan. Se suele utilizar en los bordes para suavizarlos o crear curvas.

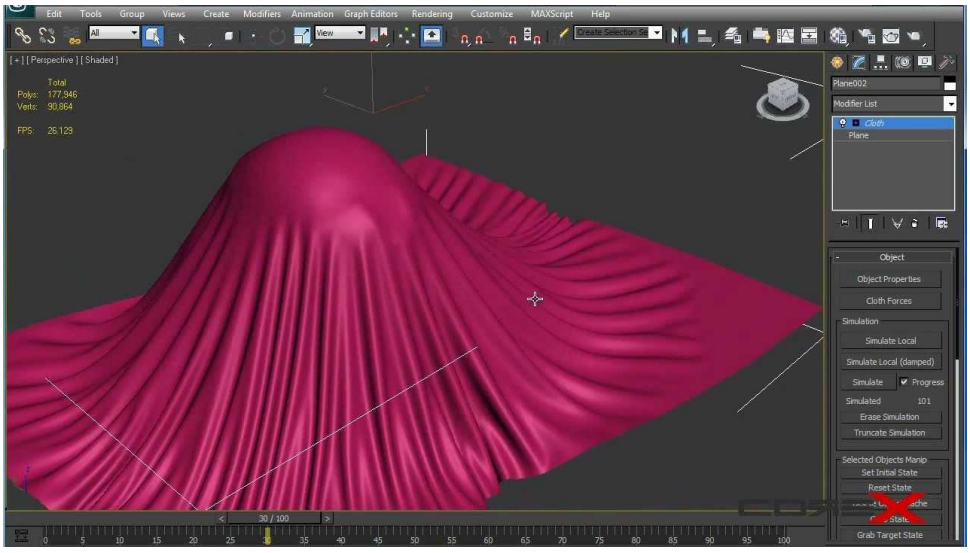


Por otro lado, tenemos los **modificadores**, que se encargan normalmente de cambiar la forma del objeto y se pueden aplicar tantos como quieras. Algunas características de estos son las siguientes:

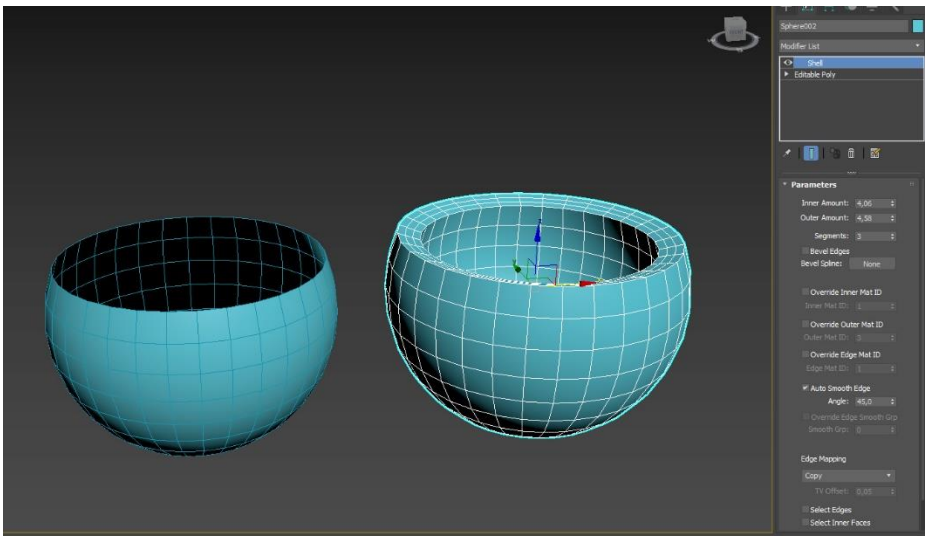
- Se pueden aplicar tantos como quieras al objeto.
- Cuando se elimina un modificador se pierden los cambios que realizó sobre el objeto.
- Cada modificador afecta a los siguientes, por lo tanto, el orden en que los añades es muy importante.
- El efecto que hace el modificador depende en gran medida de la topología que hayamos conseguido crear primero para el objeto.

Alguno de los modificadores que más he utilizado son: **Cloth**, **Shell**, **TurboSmooth**, **FFD 4x4x4** y **Unwrapp UVW**.

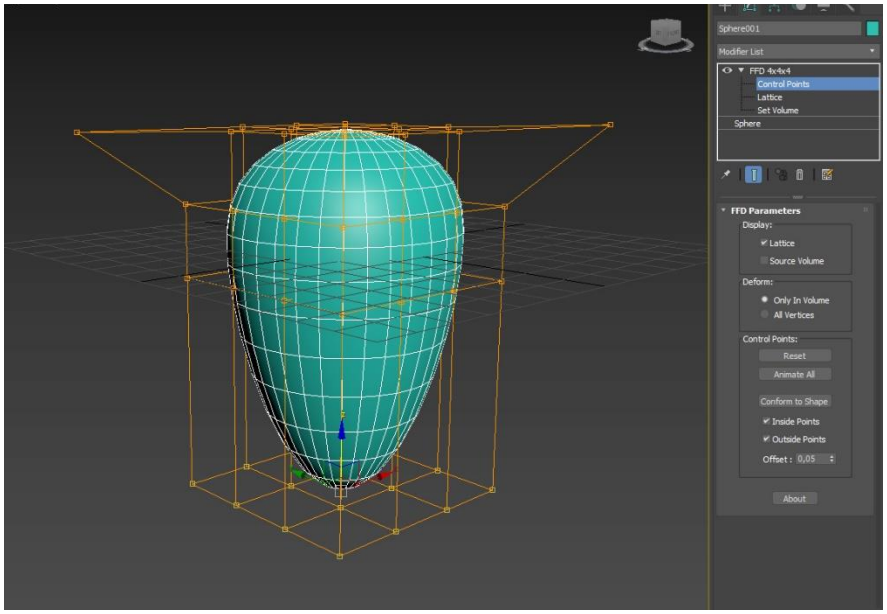
- **Cloth.** Este modificador te permite crear objetos tales como cortinas, mantas, ropa o cualquier tipo de tejido. Funciona mediante una simulación en la que dices qué tipo de tejido quieres que sea el objeto y con cuáles colisiona.



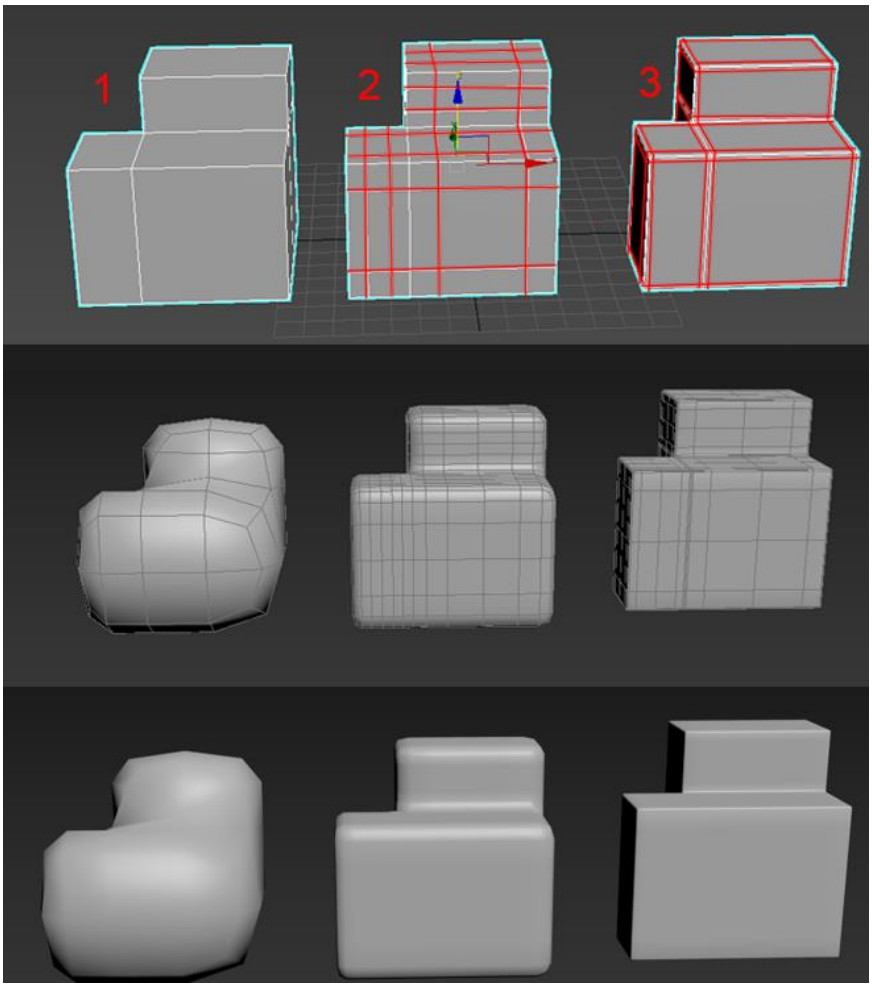
- **Shell.** Este modificador se encarga de crear una especie de caparazón cuando tienes solo una capa de polígonos, es decir, si tenemos por ejemplo un plano, le daría grosor.



- **FFD 4x4x4.** Nos permite deformar la malla mediante unos puntos de control, permite mover, rotar y escalar. Se suele utilizar cuando hay muchos polígonos como en el caso de la foto, y se quiere deformar la malla de manera más suave, como si fuese plastilina donde los polígonos más cercanos a los puntos de control se afectarán más que los más alejados.



- **TurboSmooth.** Lo que consigues con este modificador es subdividir el objeto y conseguir curvas o suavizados en la geometría del objeto. Para conseguir un buen resultado al hacer uso de TurboSmooth es necesario tener muy claro cómo funciona para realizar anteriormente una buena topología del objeto, si no el objeto se deformará y se romperá.



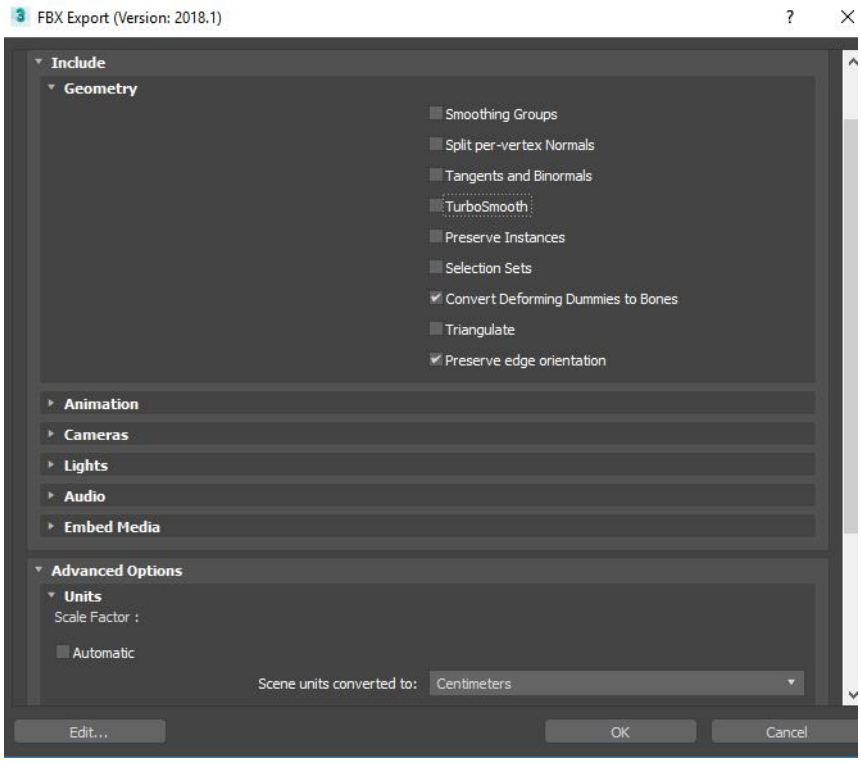
Como se puede observar en el primer caso, la malla se deforma de mala manera al aplicarle TurboSmooth, esto se debe a que no hemos tratado la topología de forma correcta. Lo que debemos hacer para conservar la forma y suavizar la geometría es crear líneas paralelas a los bordes y el modificador creará las curvas respecto a los bordes y a las nuevas líneas creadas.

En el caso dos, ponemos líneas paralelas no muy cercanas al borde, y como resultado vemos cómo los bordes quedan redondeados mientras que en el caso 3 los bordes están muy definidos debido a que hemos creado las líneas muy cercanas al borde.

- **Unwrapp UVW.** Este modificador sirve para crear el mapa de coordenadas UV para poder texturizar el objeto. Más adelante se explica a fondo como se utiliza.

Exportar modelos

Para poder texturizar los modelos en Substance Painter y en Unreal Engine 4 tenemos que exportarlos a formato ".fbx". Para ello nos dirigimos a "File" → "Export" → "Export Selected" y nos aparecerá la ventana siguiente:

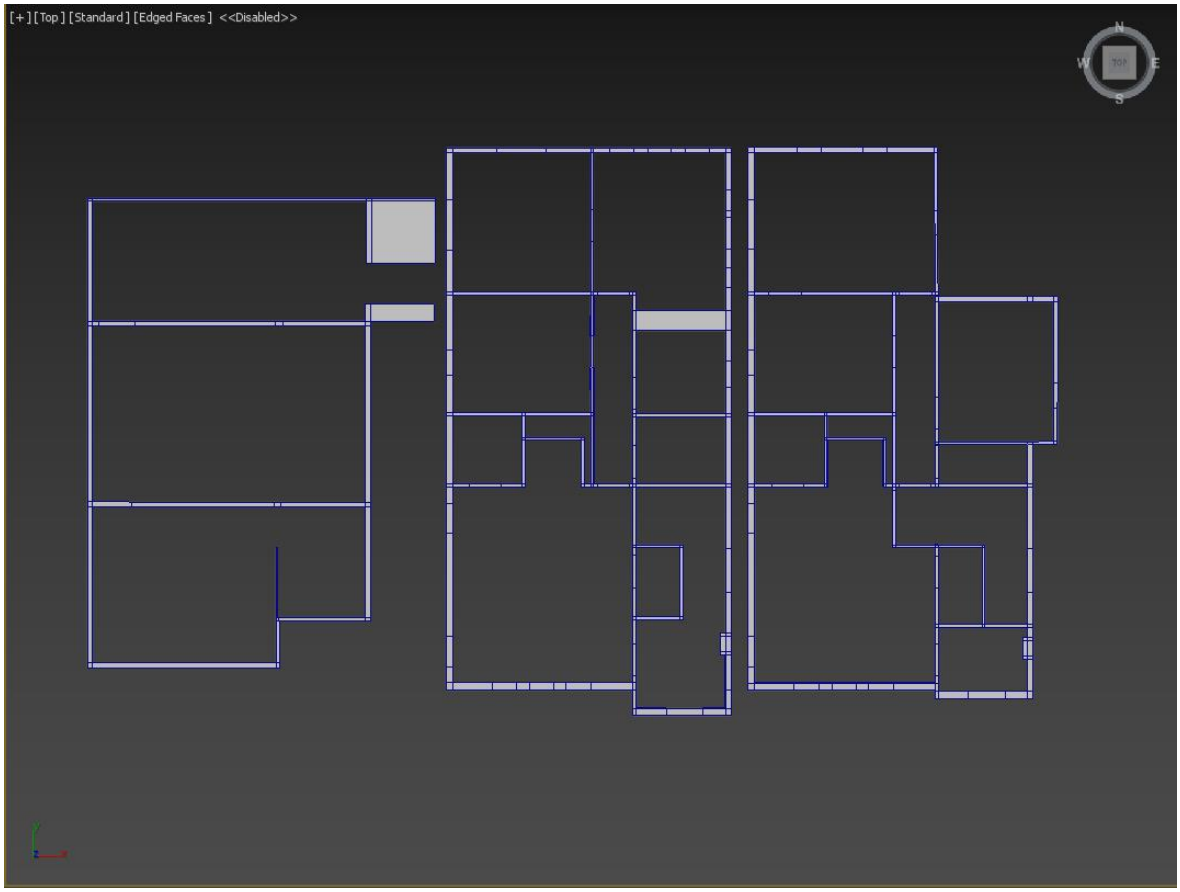


Para exportarlo correctamente, hay que tener marcado lo mismo que se ve en la imagen. Es muy importante tener desactivada la casilla de "Smoothing Groups" y "Turbo Smooth" porque si no lo marcamos estos no se conservarán, aunque pueda parecer contradictorio, ni en Unreal ni en Substance Painter. Por último, hay que asegurarse de que lo tenemos en la unidad de medida adecuada. Como no tengo ninguna animación, luces o similares, dejaré el resto de desplegables por defecto.

6.3.1.1.2 Resultados Obtenidos

A continuación, mostraré algunos de los objetos que he modelado en 3dsmax (no están todos, pero si la mayoría y los más grandes) divididos y ordenados por algunas de las habitaciones de la escena. También cabe decir que la gran mayoría de estos objetos que enseño son Low Poly, necesarios para no sobrecargar la escena y obtener la mejor optimizacion posible, pero

también se tuvo que realizar una versión de los mismo objetos en High Poly para posteriormente hacer *backing* y obtener los mapas de normales en la fase de texturizado.



Este es el plano de la casa que dibujé anteriormente mediante líneas en 3dsmax y al que gracias al extrude le di volumen. En un principio se contempló la idea de hacer paredes procedurales, pero puesto que nuesto objetivo era hacer esa casa específica y no necesitaba ser ampliada en el futuro ni hacer nuevos escenarios se descartó la idea de hacerlo así.

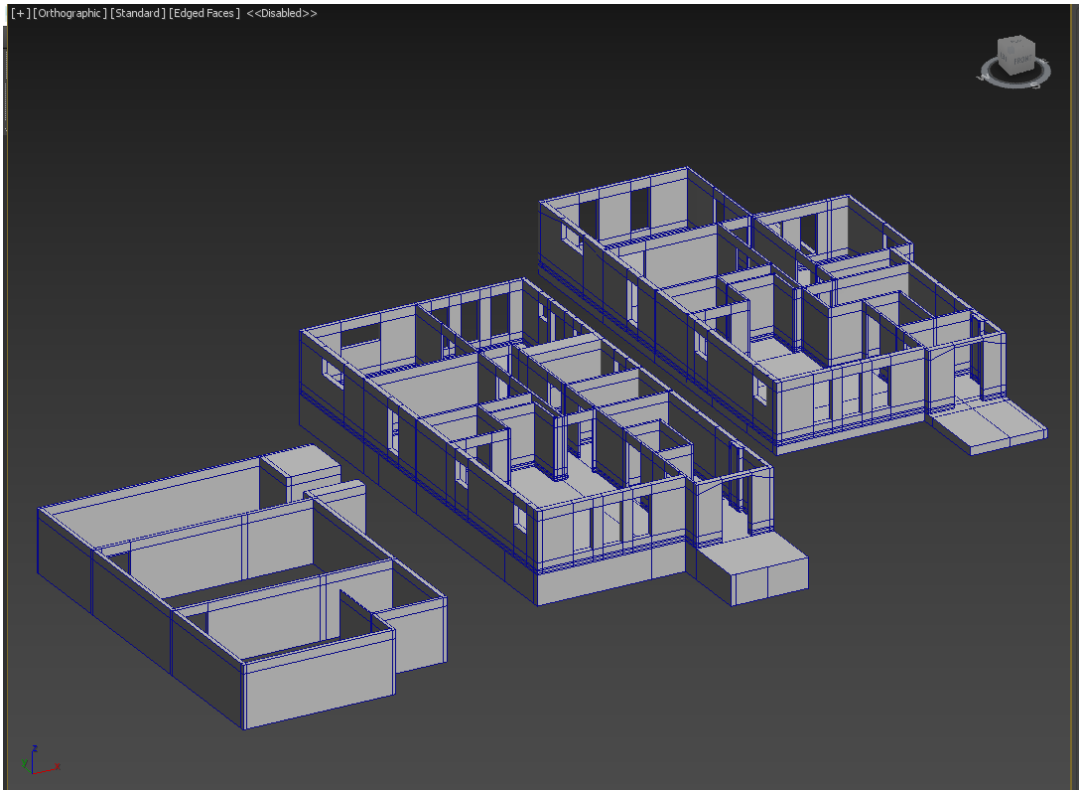


Ilustración 19.Modelado estructura de las tres plantas de la casa

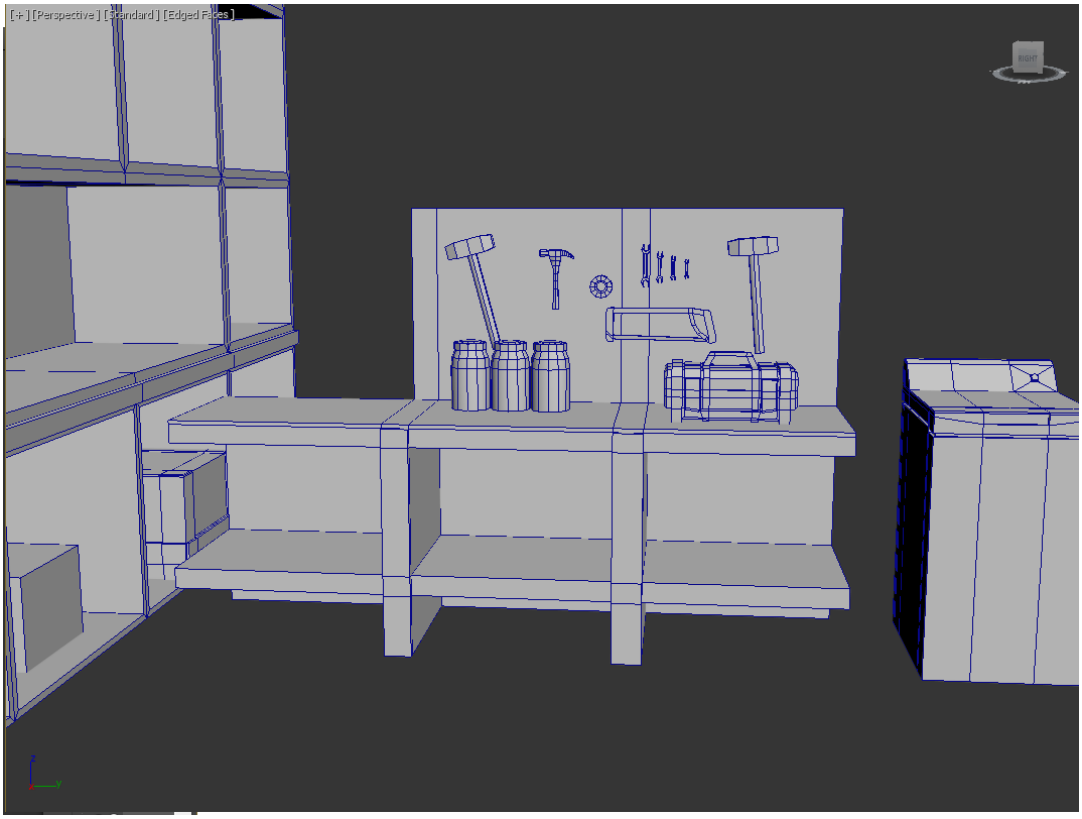


Ilustración 20.Modelado de los objetos del taller

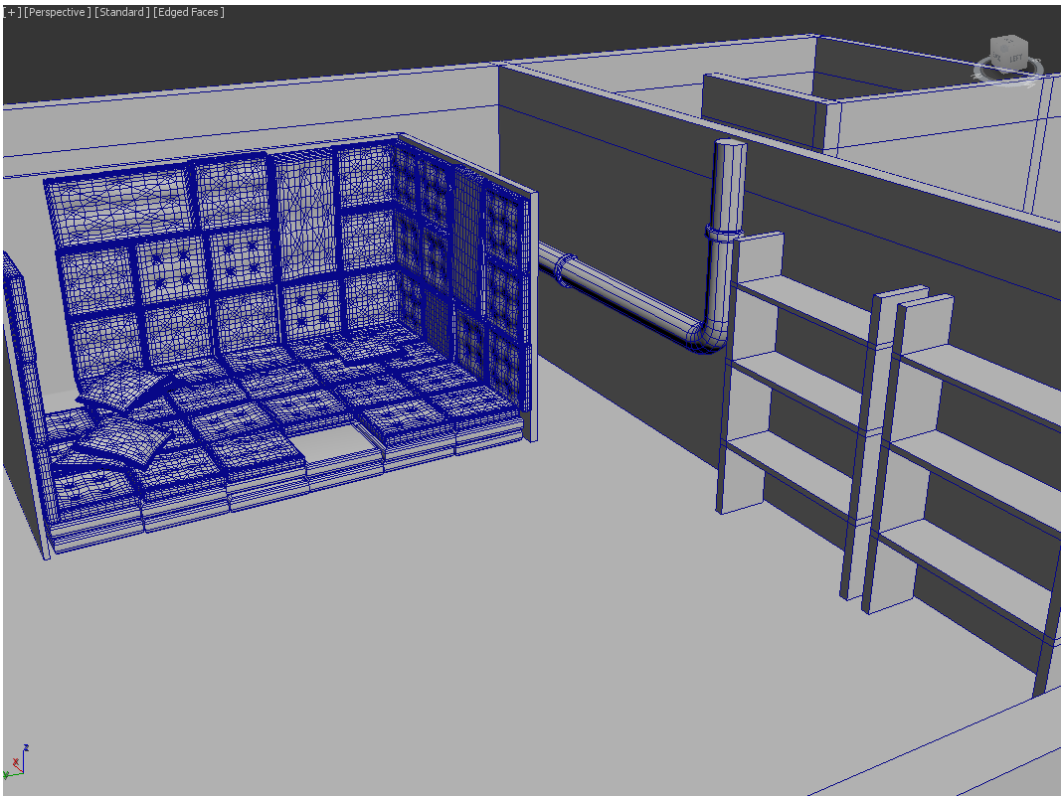


Ilustración 21. Modelado de los objetos del sótano

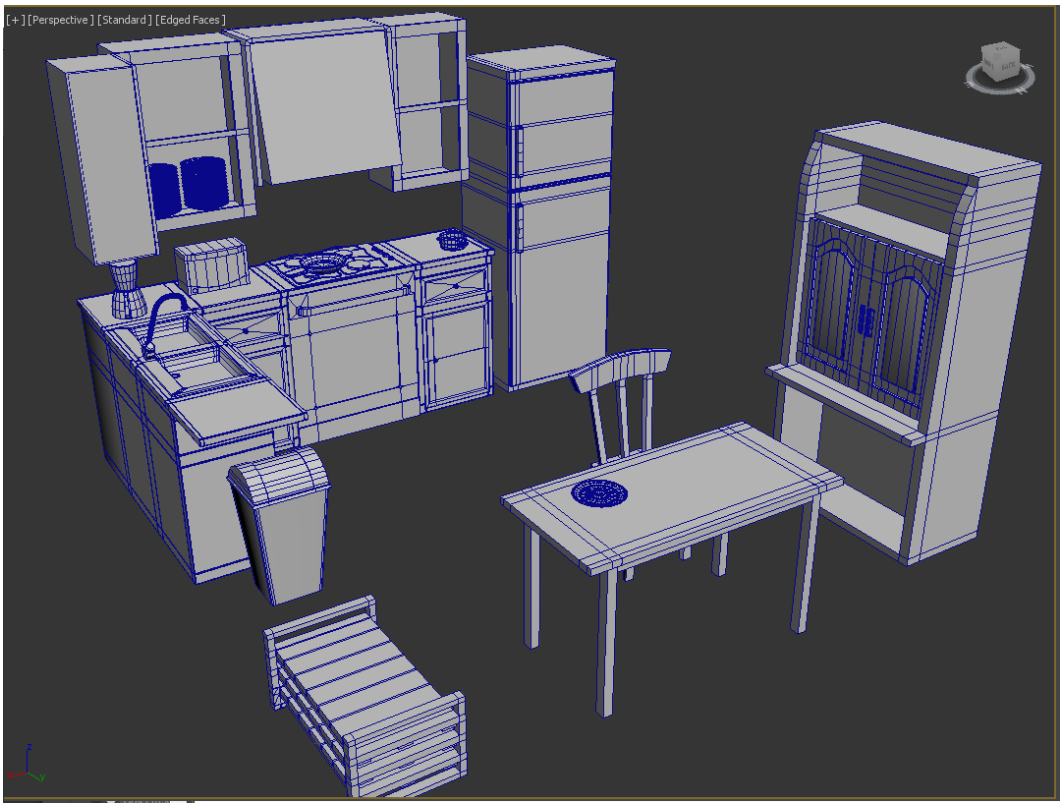


Ilustración 22. Modelado de los objetos de la cocina

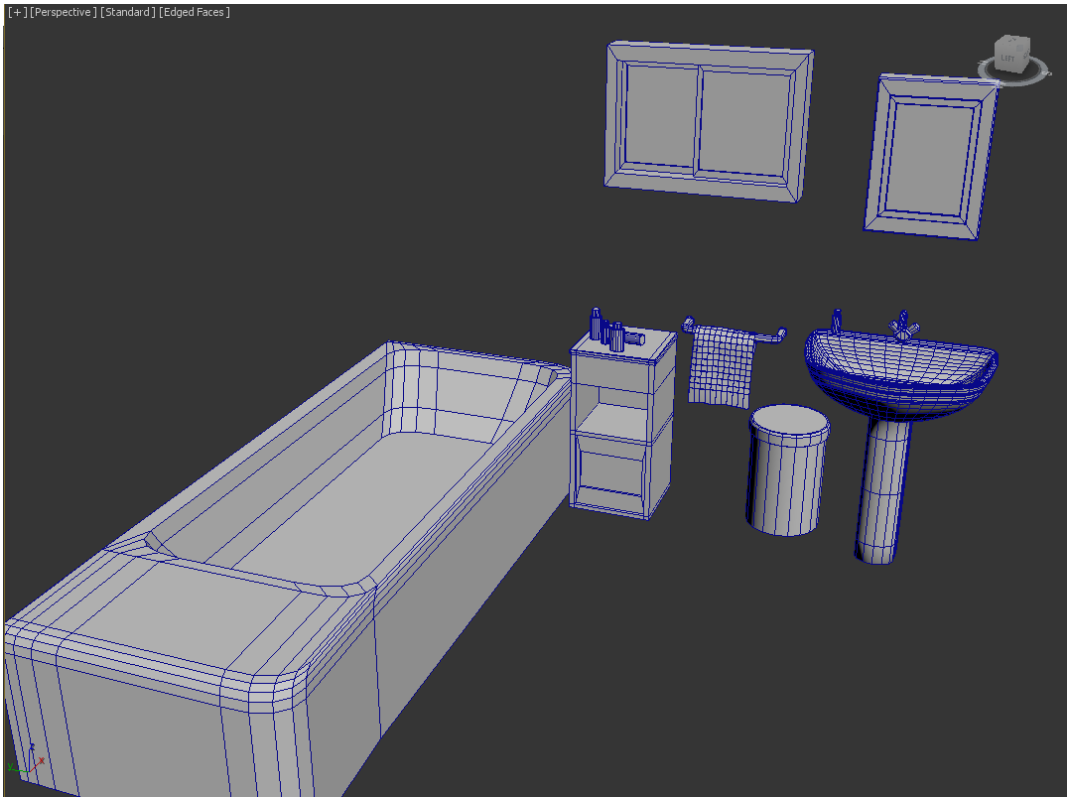


Ilustración 23. Modelado de objetos del baño

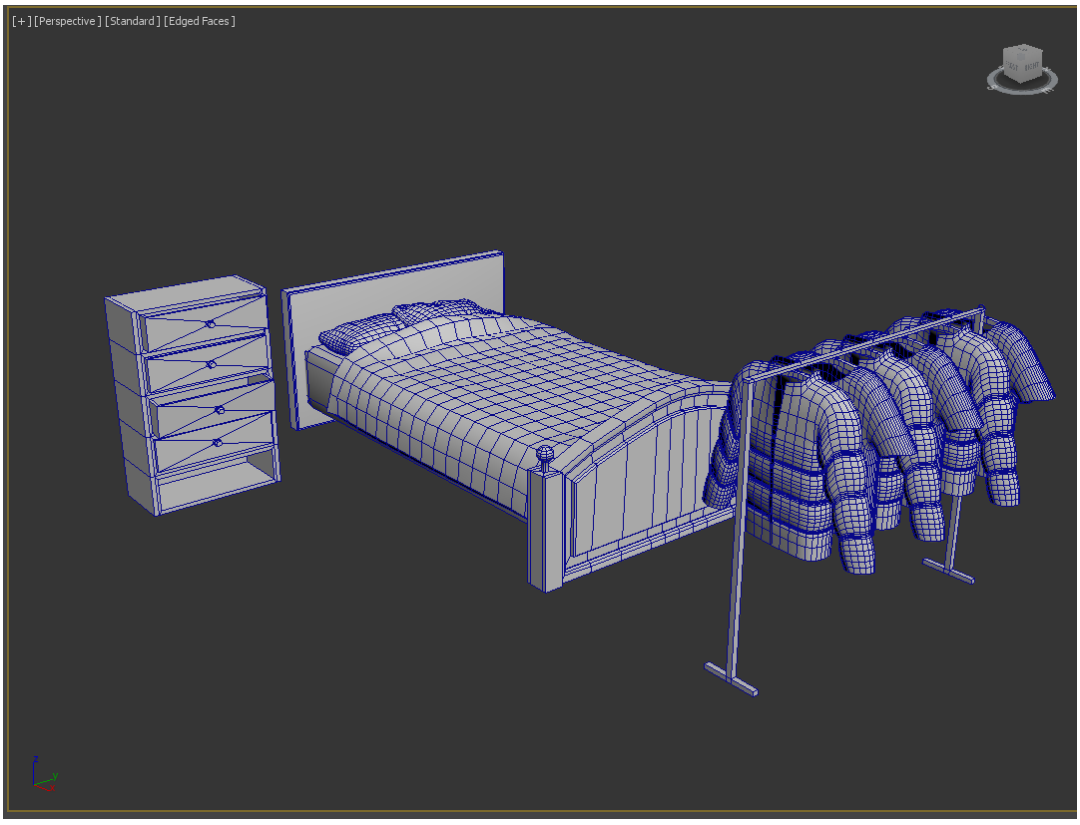


Ilustración 24. Modelado de objetos de la habitación

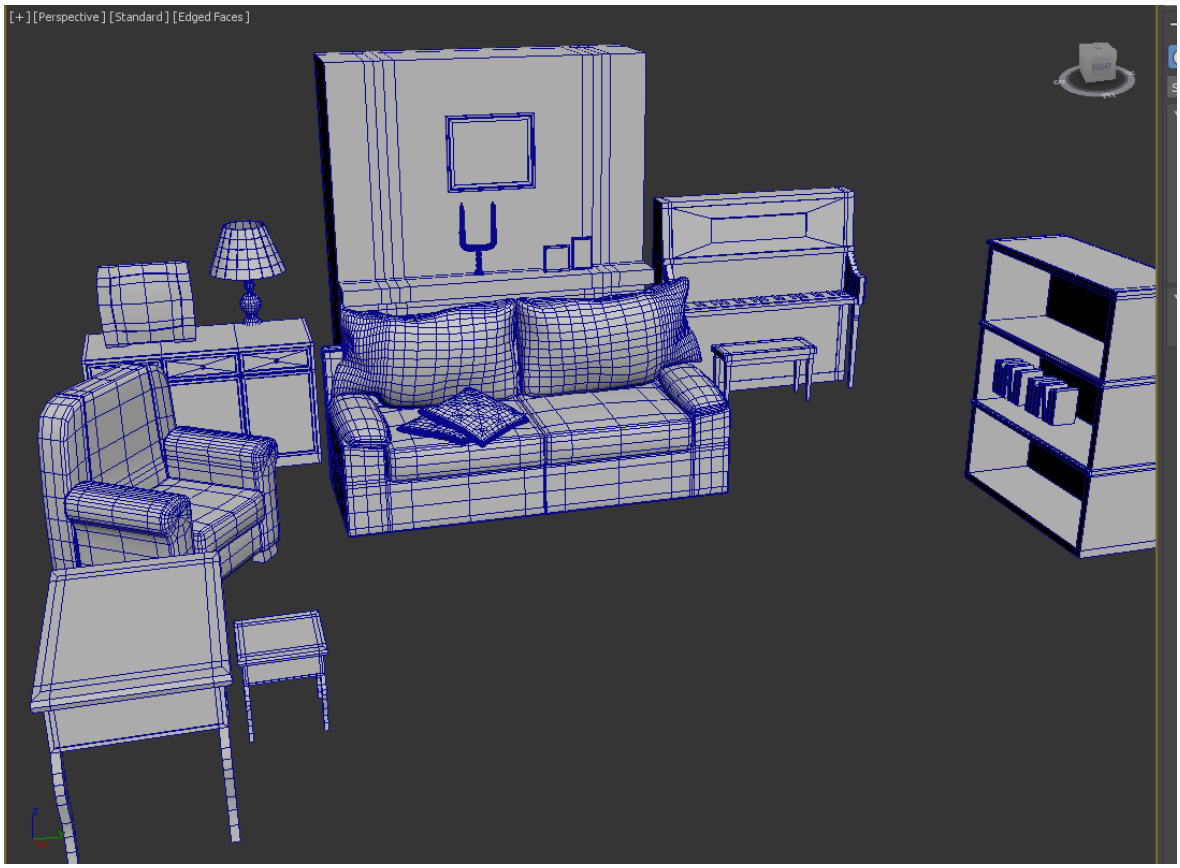
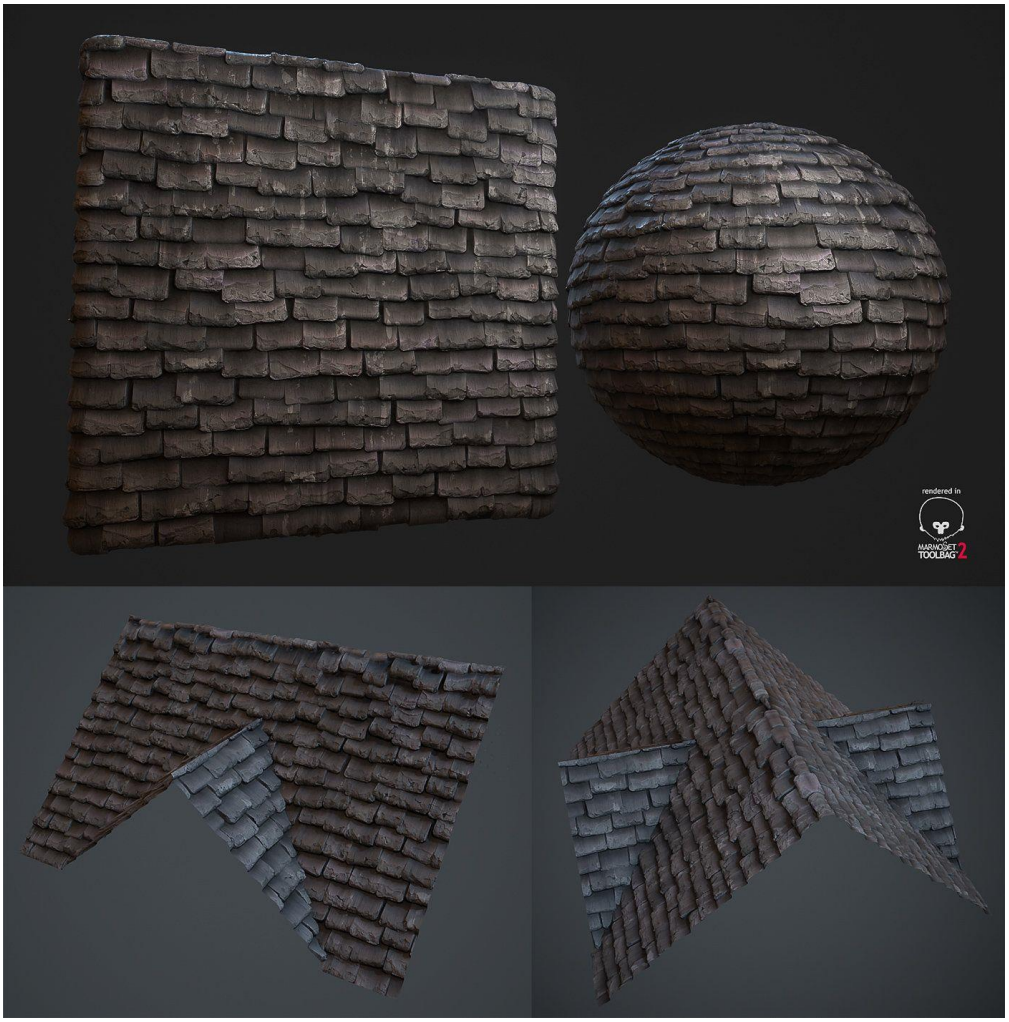


Ilustración 25. Modelado de objetos del salón

6.3.1.2 Mapa de texturas UV

Una vez creado el modelo, antes de empezar a crear las texturas hay que tener en cuenta una cosa, si debemos usar **texturas procedurales** o no. Si queremos un objeto con texturas uniformes que no dependan de la posición, es decir, si creamos una pared que posee dibujos que se repiten, por ejemplo, ladrillos, no será necesario crear los mapas de texturas, ya que cuando añadas ese material al objeto este se distribuirá por toda la malla de manera procedural o repetida por toda la superficie de manera correcta.

En la imagen siguiente se aprecia lo que sería una textura procedural. Cabe decir que mediante el uso de parámetros puedes decir la cantidad de veces que se repite la textura sobre la malla; obviamente, si aumentas la cantidad de repeticiones se verán más pequeños los detalles.



Las texturas Procedurales constituyen una buena manera de texturizar un modelo. Lo que es realmente bueno de este tipo de texturas es que siempre "calzan", se ajustan perfectamente al modelo. Las imágenes 2D, en cambio, no siempre se ajustarán tan bien a un modelo complejo. Crear texturas procedurales es relativamente sencillo, y ofrecen una manera rápida de obtener buenos resultados. Sin embargo, hay situaciones en las que este tipo de texturas no es suficiente. Por ejemplo, la piel de una cabeza humana nunca se verá lo suficientemente bien cuando es generada proceduralmente. Las arrugas en un rostro humano, o las saltaduras de pintura en un automóvil no aparecen en lugares al azar, sino que dependen de la forma del modelo y de su uso. Las imágenes pintadas manualmente le dan al artista control total sobre el resultado final. Un mapa UV es una forma de asignar la parte de una imagen a un polígono en el modelo. Cada vértice del polígono es asignado a un par de coordenadas 2D que definen qué parte de la imagen es mapeada. Estas coordenadas 2D se llaman UV (comparar con las coordenadas XYZ en 3D). La operación de crear estos mapas UV se

conocen también como "despliegue" (*unwrap* en inglés), debido a que todo ocurre como si la malla fuera desenvuelta o desplegada sobre un plano 2D.

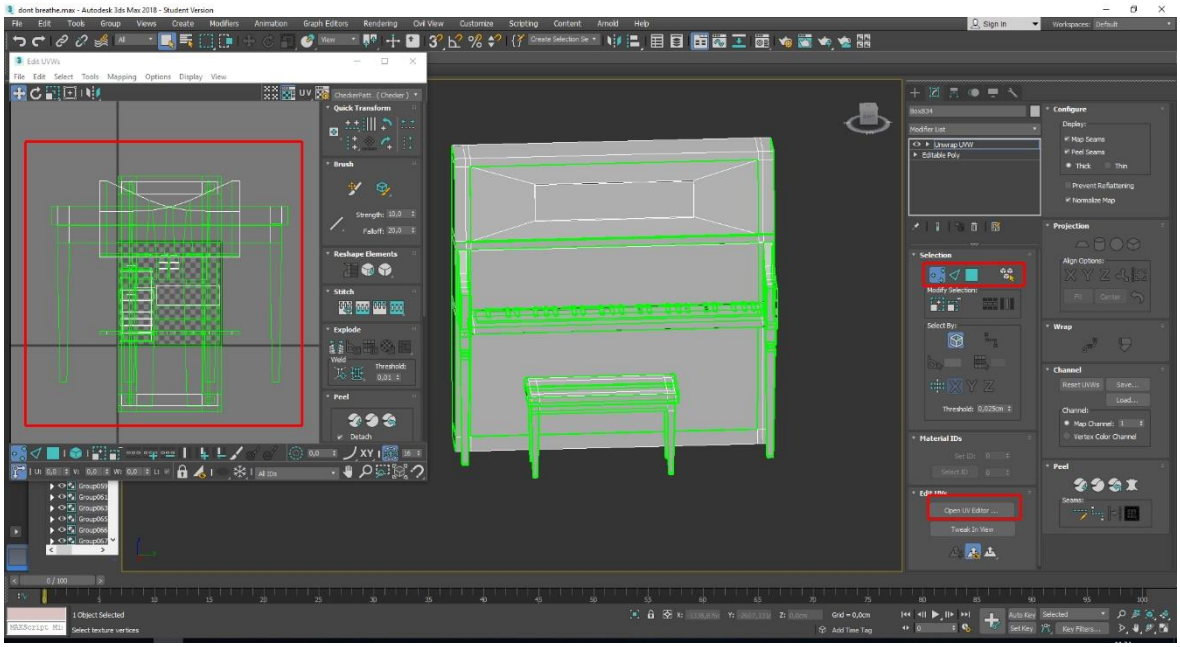
Unwrap UVW

Este es un modificador que utilizaremos para crear mapas de texturas complejas (texturas no procedurales) y hacer que se adapten perfectamente las imágenes 2D al objeto 3D.

Mediante el uso de este modificador he podido prácticamente texturizar todos los objetos que componen la escena.

A continuación, explico cómo es el proceso de mapeado mediante este modificador. Para ello, vamos a ver como ejemplo el modelo del piano. Para empezar, si tuviéramos un modelo en High Poly y otro en Low Poly para hacer *backing* y recuperar detalles, el mapeado se hace sobre el modelo Low Poly. Para ello tenemos que aplicarle el modificador Unwrapp y nos aparecerá el menú de la siguiente imagen.

Teniendo todas las caras seleccionadas (estas aparecerán en rojo), le damos a la opción “Open UV editor” y nos aparecerá una nueva ventana donde aparecen con líneas verdes representados los polígonos de la malla. Estas líneas están todas acumuladas y superpuestas dentro de un recuadro que será la imagen o textura que se pondrá sobre el objeto.



Lo que hay que hacer es separar esas líneas verdes por grupos para que no se superpongan y así poder texturizar luego de manera sencilla. Para ello podemos hacer dos cosas:

Por un lado, si queremos ser muy estrictos y crear unas UV perfectas y que luego no haga nada raro, como que se corten partes de la imagen de repente, tenemos que ir dividiendo la maya a nuestro gusto. Para hacer una división especifica de la malla, lo que tenemos que hacer será seleccionar en el objeto las caras que queremos dividir y darle a "Break". Esto lo que hace es separar los polígonos seleccionados del resto de la malla. A continuación, le damos a "Mapping" y te saldrán diferentes opciones que mapearán dependiendo de la orientación de las caras. Luego, eso lo mueves fuera de la textura y, una vez tengas hecho eso con todas las caras, puedes moverlo y escalarlo para tenerlo todo ordenado.

En la siguiente foto hemos dividido dos veces, una con la parte frontal y otra en el lateral. Estos son los resultados aplicándole un "Normal mapping" en forma de caja. Luego haríamos lo mismo con las caras restantes.

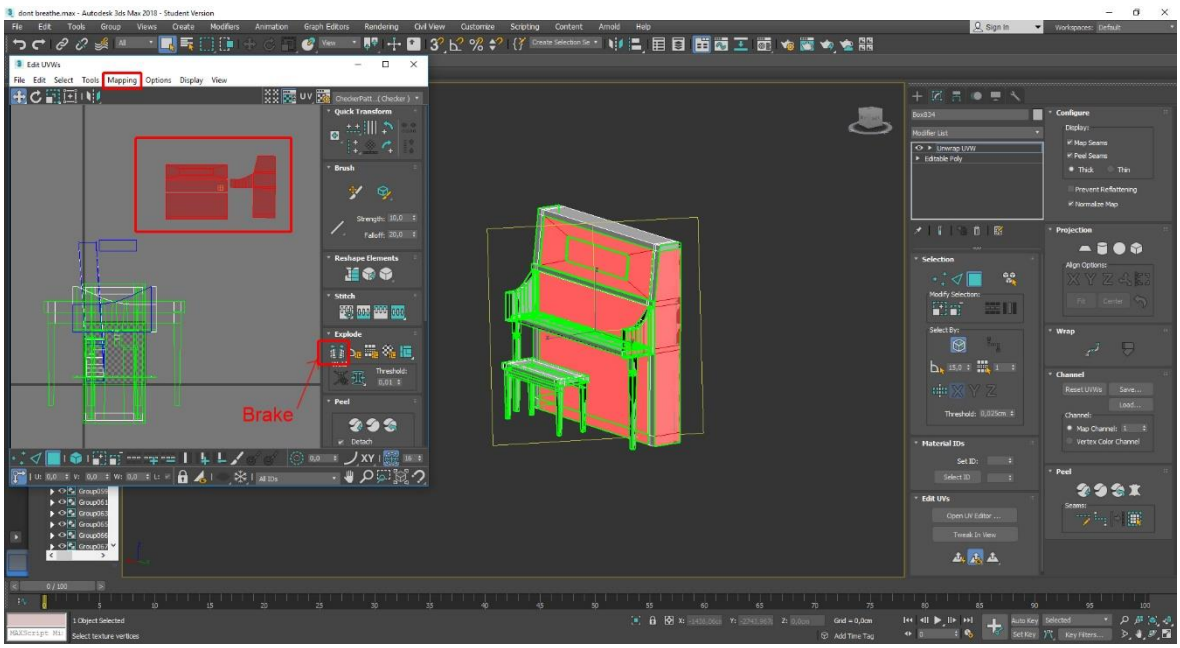
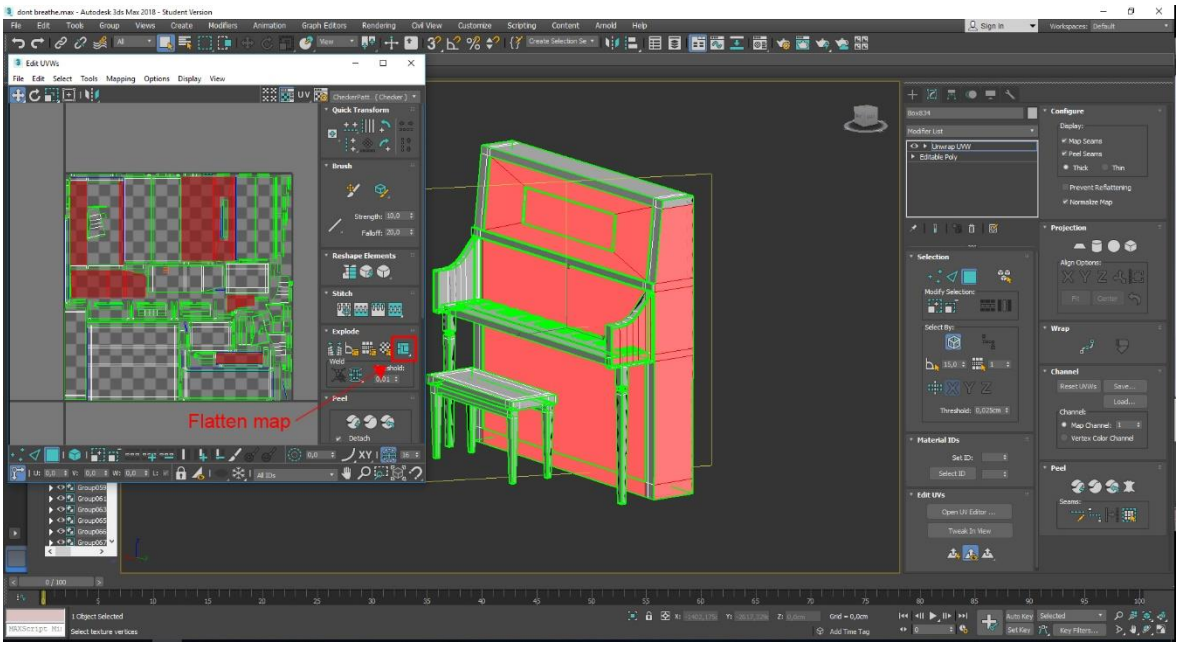


Ilustración 26. Mapeado mediante modificador Unwrap UVW

Por otro lado, para hacer una división más automática pero no tan perfecta, simplemente tenemos que dirigirnos con todos los polígonos seleccionados a "Mapping" y darle a "Flatten Map", que divide todo automáticamente en relación con la superficie y orientación de los polígonos del modelo.



Como se puede observar, aquí ya no se superponen. Unwrap UVW lo divide y lo despliega todo para que se vea plano, pero si seleccionamos las mismas caras que en el primer ejemplo, se ve cómo esas caras que hemos seleccionado no están agrupadas, y haría que el texturizado no sea muy bueno.

Si vamos a realizar la textura en Photoshop, necesitaremos unas UV perfectamente creadas por zonas, por lo que tendremos que hacer la primera opción. Si por el contrario queremos texturizar en Substance Painter, con la segunda opción automática bastaría, ya que en Substance vas pintando sobre el objeto y no sobre la imagen en 2D.

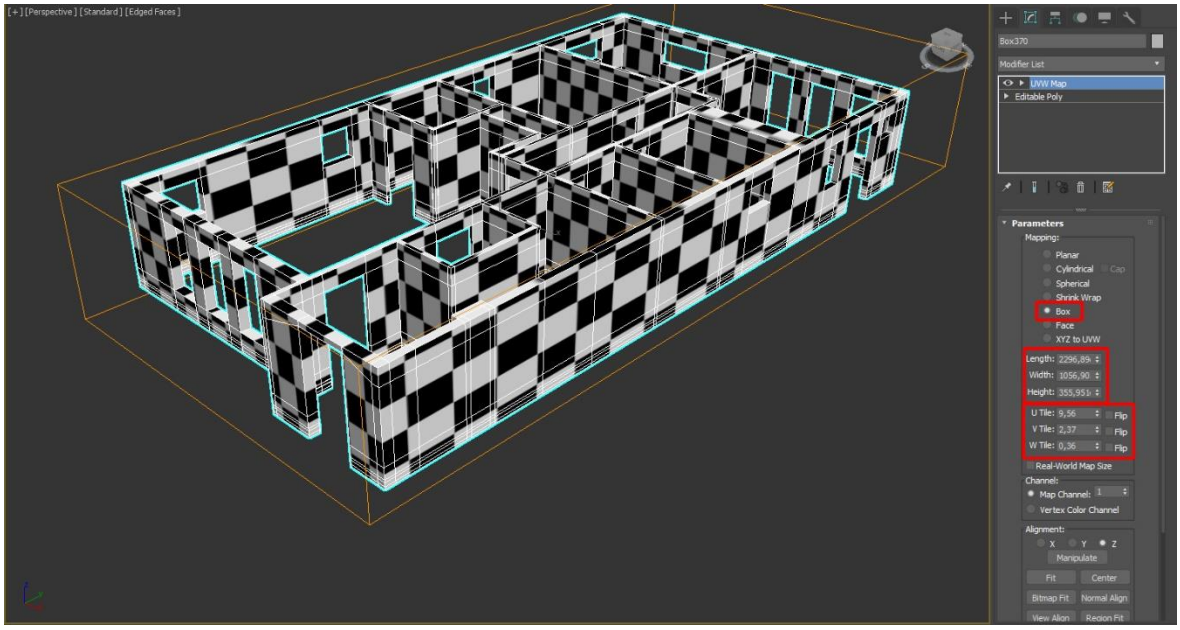
UVW map

Este modificador, a pesar de no ser necesario, lo usaremos para cuando queramos hacer uso de texturas procedurales. Mediante los parámetros que ofrece haremos que se adapte mejor la textura a nuestro modelo.

Particularmente yo este modificador lo he usado para el texturizado de las paredes, porque eran objetos muy grandes y necesitaría muchísima resolución para poder hacerlo.

Este modificador te permite crear formas geométricas como esferas, cubos, cilindros, etc., entonces creará el mapa de UV respecto a las caras de esas geometrías y superponiendo las UV de las caras interiores. Por ejemplo, las paredes de la casa tienen muchas caras y puesto que no podemos crear una textura dividiendo todas las caras y meterlas en una sola imagen

de textura porque necesitaría mucha resolución, lo que haremos será crear las UV superpuestas que harán que la imagen procedural se repita por todas las paredes de manera uniforme. En el caso de las paredes, que tienen forma cúbica, le pondremos la forma de cubo en la interfaz y el tamaño, y a continuación le diremos la cantidad de veces que queremos que se repita la textura procedural (por ejemplo, un grupo de ladrillos) que le aplicamos sobre ella con las opciones de *tiling*.



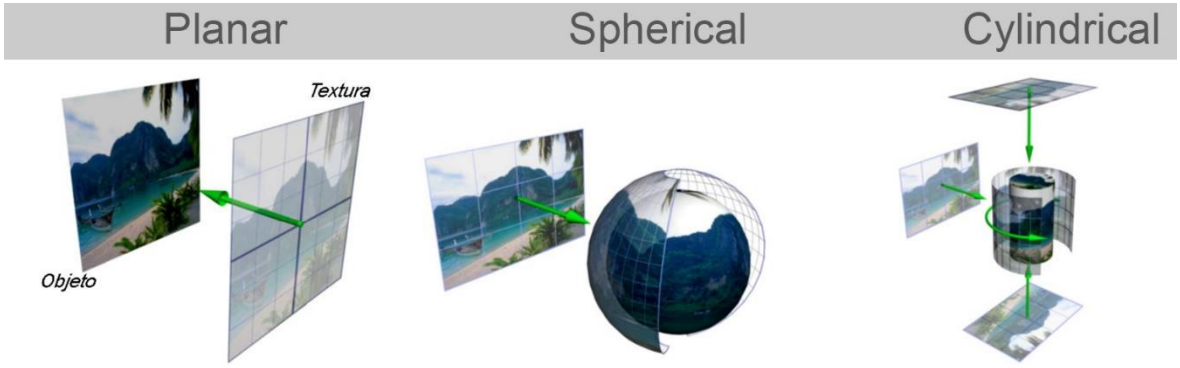


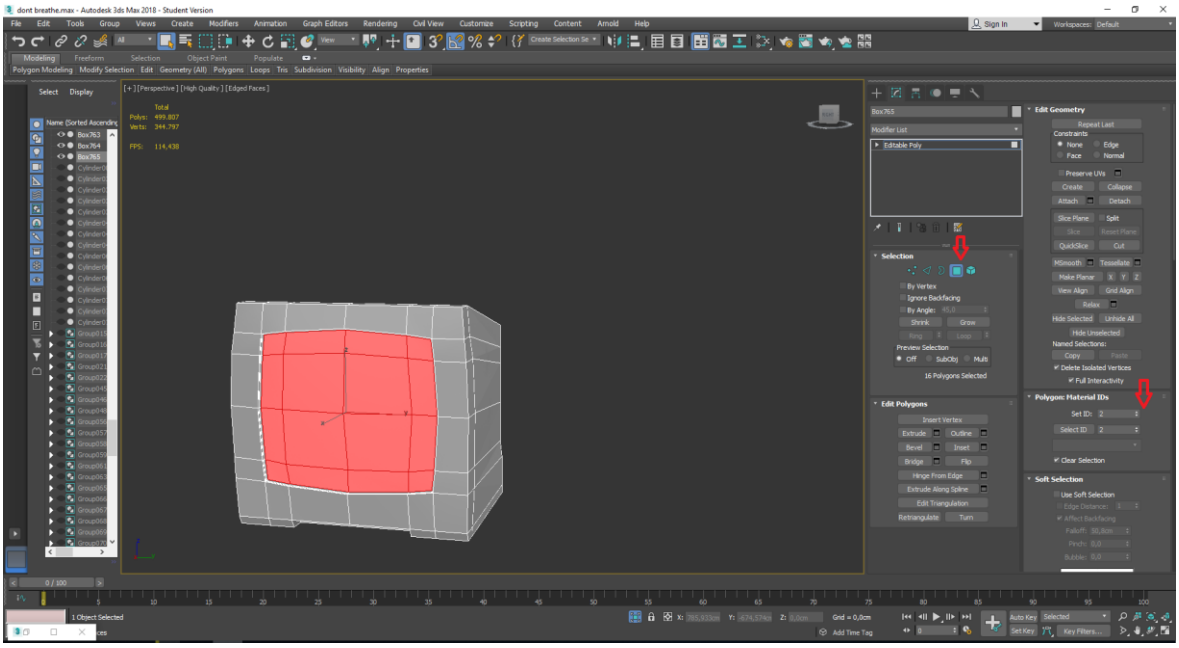
Ilustración 27. Mapeado mediante modificador UVW map según geometría

Diferentes texturas para un mismo objeto y *tiling*.

A veces, nos interesa que un mismo objeto tenga diferentes materiales. Por ejemplo, en un cuchillo nos interesa tener el mango de plástico y la hoja metálica. Para ello hay diferentes técnicas que nos permiten llevar a cabo dicha acción.

Mediante ID y material “Multi/Sub-Object”.

En 3dsmax, cuando tenemos ya el modelo 3D finalizado podemos, dentro del modo “Editable poly”, seleccionar las caras del objeto que queramos para cada material y le pondremos una **id** específica. Por ejemplo, si tenemos el modelado 3D de una tele, nos interesa dividir por un lado la pantalla y por otro la estructura de plástico. Nos dirigiremos a la opción de selección por caras y seleccionamos primero la estructura y le asignamos la id 1 y a continuación seleccionamos las caras de la pantalla y le pondremos la id 2.



Por último, vamos a ponerle al objeto el material que nos permite asignarle múltiples materiales. Para ello, nos dirigimos al menú de los materiales y ponemos el modo Slate material editor (1), y buscamos en el buscador el nombre Multi/Sub-object (2). Arrastramos a la pantalla de nodos y aparecerá un material con muchos números. Esos son las id (3) a las cuales les vamos a asignar el material (4). Le ponemos dos colores bastantes diferenciados para que se vea el resultado y finalmente le damos a asignar el material al objeto seleccionado (5).

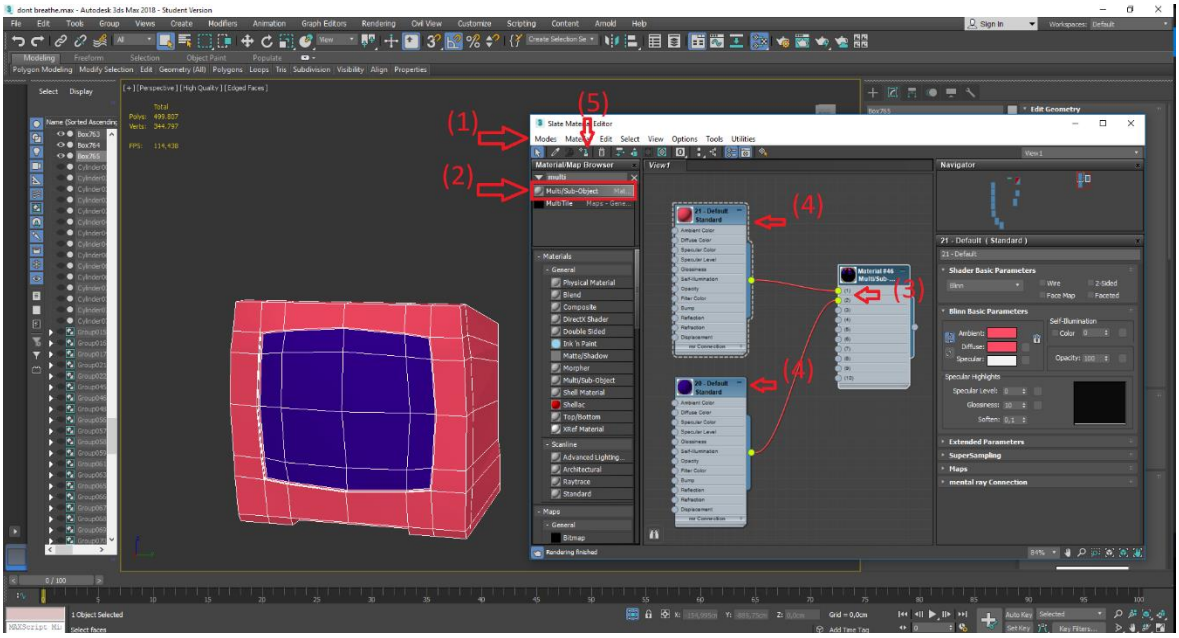
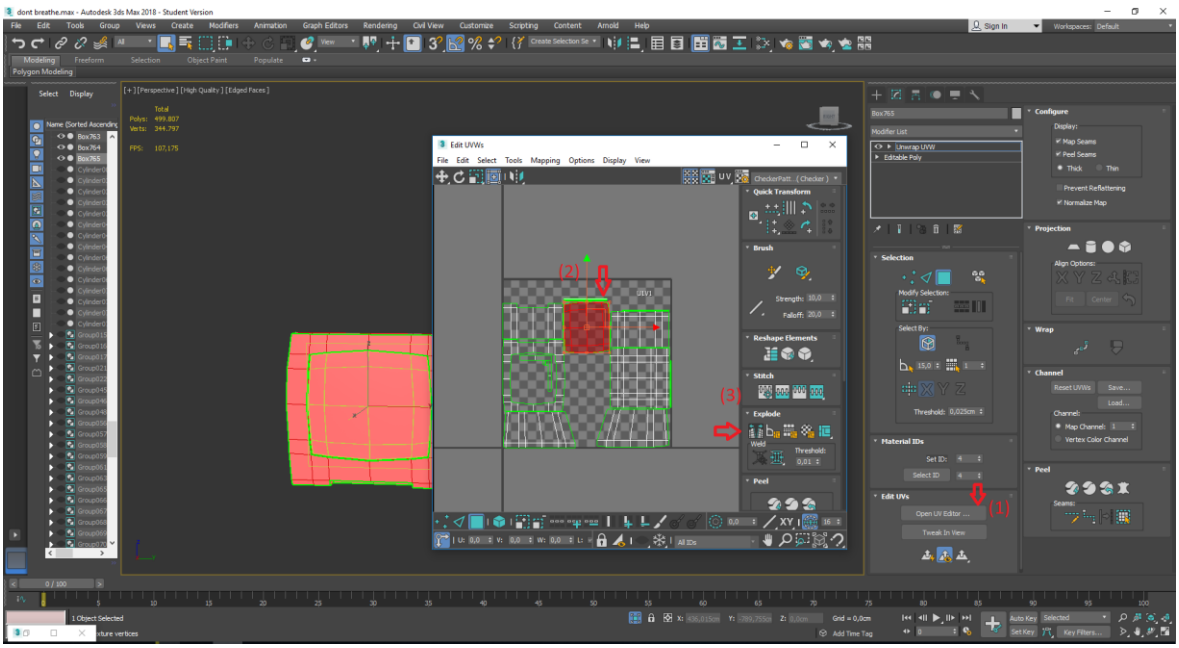


Ilustración 28. Material multitile.

Mediante Multi-Tile con el modificador Unwrap UVW

Otra manera de crear diferentes materiales para un mismo objeto es crear diferentes mapas de texturas o Tiles, ya explicado anteriormente. A esto se le llama Multi-Tile, y creará diferentes mapas con UV. Después añadiremos a cada mapa los materiales y texturas que deseemos y, al aplicárselo al objeto, él mismo reconocerá donde va cada material.

Para hacer esto, lo único que tenemos que hacer será meternos dentro del modificador Unwrapp UVW y abrir el editor de UV (1). Una vez ahí, seleccionamos los polígonos que forman la pantalla (2) y los separamos del resto de polígonos dándole a la opción “break” (3).



Por último, movemos esos polígonos fuera del mapa y se creará otro aparte, y redimensionamos para que se ajuste a ella (4).

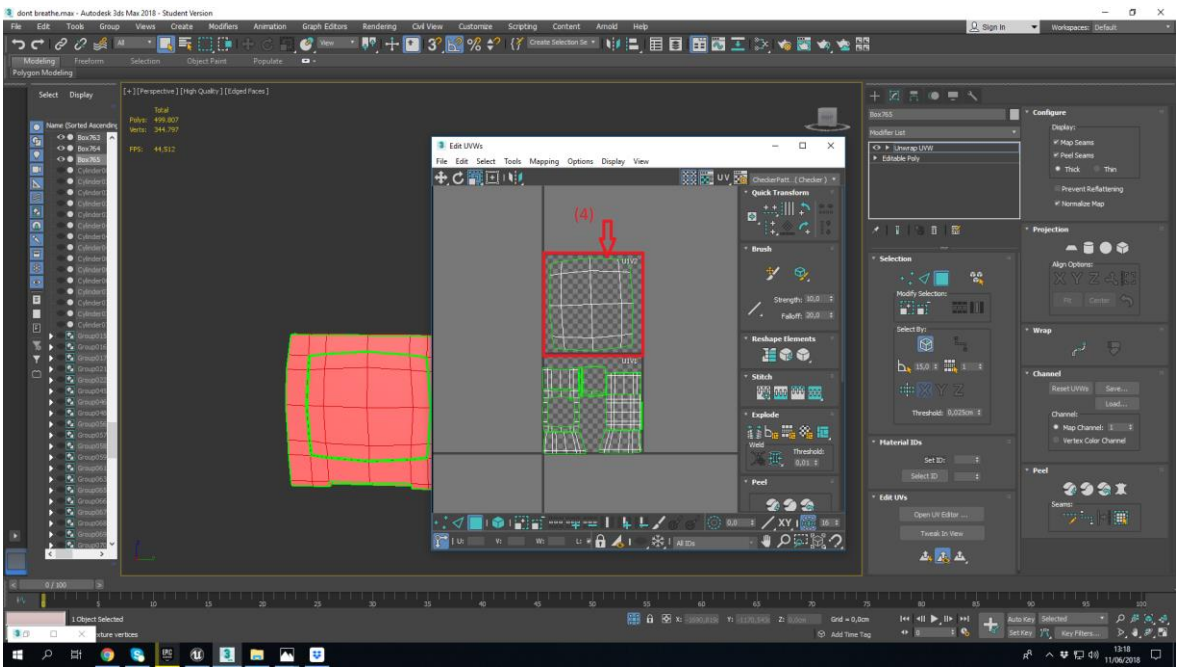


Ilustración 29. Diferentes mapas de texturas para un mismo objeto mediante multitiles

De ambas formas, tanto Unreal Engine 4 como Substance Painter detectarán los diferentes mapas y nos permitirán tratarlos por separado.

6.3.1.3 Creación de materiales y texturas.

¿Qué tipos de mapas vamos a utilizar para nuestros materiales?

Anteriormente ya hemos hablado sobre qué son los materiales, en este apartado hablaremos sobre cómo crearemos estos y cómo obtener las características del material que deseemos simular mediante sus parámetros.

Antes de empezar voy a nombrar y explicar los principales tipos de mapas de los que estará compuesto el material para situarnos un poco antes de entrar en materia.

Mapa difuso: coloca la imagen 2D de la textura en el modelo 3D. El uso de un mapa para el color difuso es similar a pintar una imagen en la superficie del objeto. Por ejemplo, si desea un muro de ladrillo en la escena, puede utilizar una imagen de mapa de bits con una imagen de ladrillos como mapa difuso en el material aplicado al objeto de muro.

Mapa de oclusión ambiental: los mapas de oclusión se utilizan para informar de qué partes del modelo deberían recibir alta o baja iluminación indirecta.

Se puede decir que la oclusión ambiental es un efecto de oscurecimiento que se genera en aquellas zonas de las superficies donde llega menos energía lumínica o menos rebotes de luz, típicamente las esquinas, los encuentros entre superficies, ranuras, oquedades, etc. Sin oclusión ambiental a veces las escenas tienden a mostrarse planas, poco contrastadas y con los encuentros mal definidos.

Mapa especular: los mapas especulares son los que definen el brillo y color de las partes iluminadas de un objeto, es decir, cómo y en qué medida el material es capaz de reflejar la luz que le llega. Esto es muy útil para darle al material propiedades como ser un objeto metálico o de plástico, dependiendo de la cantidad de luz que reflejen y como.

Mapa de normales: simulan la impresión de una Superficie 3D, es decir, de relieve. Pero este relieve no va a proyectar ninguna sombra y no obstruirá otros objetos. Si el ángulo de cámara es rasante en relación a la superficie, nos daremos cuenta de que la superficie no tiene relieve en realidad.

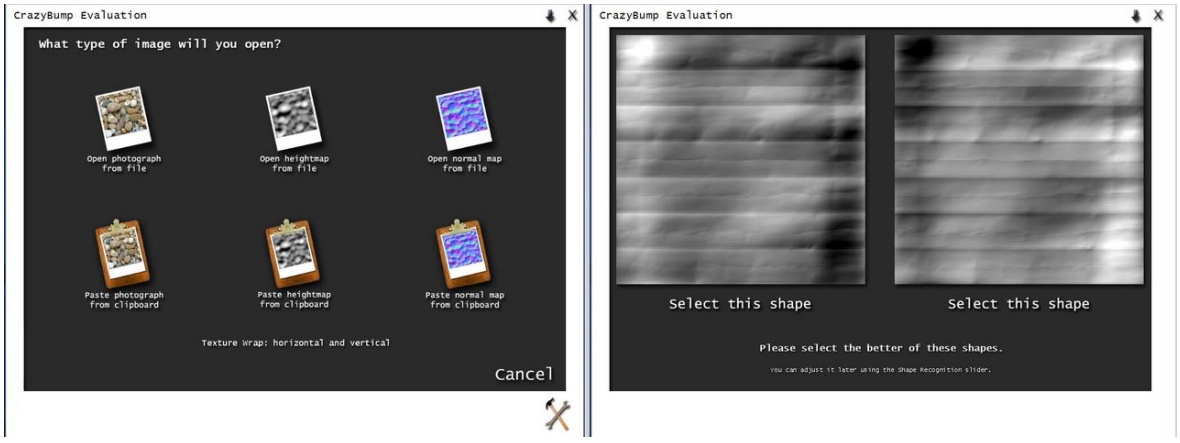
Creando texturas y materiales

En este apartado explicaré paso a paso cómo crearé ejemplos de texturas utilizando CrazyBump, Substance Painter y Photoshop y finalmente cómo se importan a Unreal Engine 4.

CrazyBump

Vamos a crear una textura procedural de madera como ejemplo que luego utilizaremos en Substance Painter. Para ello, necesitaremos una textura de madera. El mapa difuso de la madera lo obtenemos de un repositorio de internet y haremos los retoques necesarios en Photoshop, como añadir detalles, ajustar la saturación, o recortar algo que no quisiésemos.

A continuación, abrimos CrazyBump y seleccionamos importar el mapa difuso y nos aparecerán dos opciones. Elegiremos la que mejor resultado nos dé; yo elegiré el de la izquierda, que me crea los colores oscuros como hendiduras para el mapa de normales.



Una vez seleccionado el de la izquierda nos aparecerá lo de la siguiente imagen, a la izquierda se ve una previsualización de cómo se verá la textura y a la derecha tenemos parámetros que podemos modificar para los diferentes mapas (difuso, especular, oclusión ambiental, normales y desigualdad). Yo solo modifico el especular y el de normales para

obtener más o menos brillos con la especular y detalles más o menos definidos con el mapa de normales.

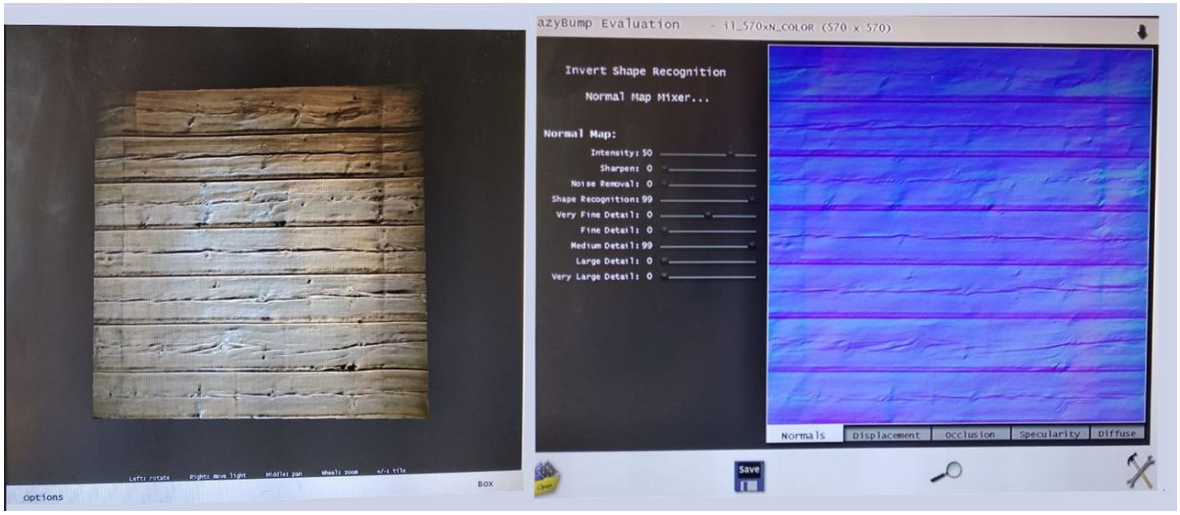


Ilustración 30. Creación de mapas de texturas en Crazy Bump

Una vez finalizado el proceso anterior y tengamos los resultados deseados guardamos todas las texturas y obtendremos los siguientes mapas:

Por último, para poder utilizar esta textura en Substance Painter, la importamos, marcando en cada uno que son texturas y dejamos la última opción marcada en “shelf-shelf” para que no desaparezca después de cerrar el programa.

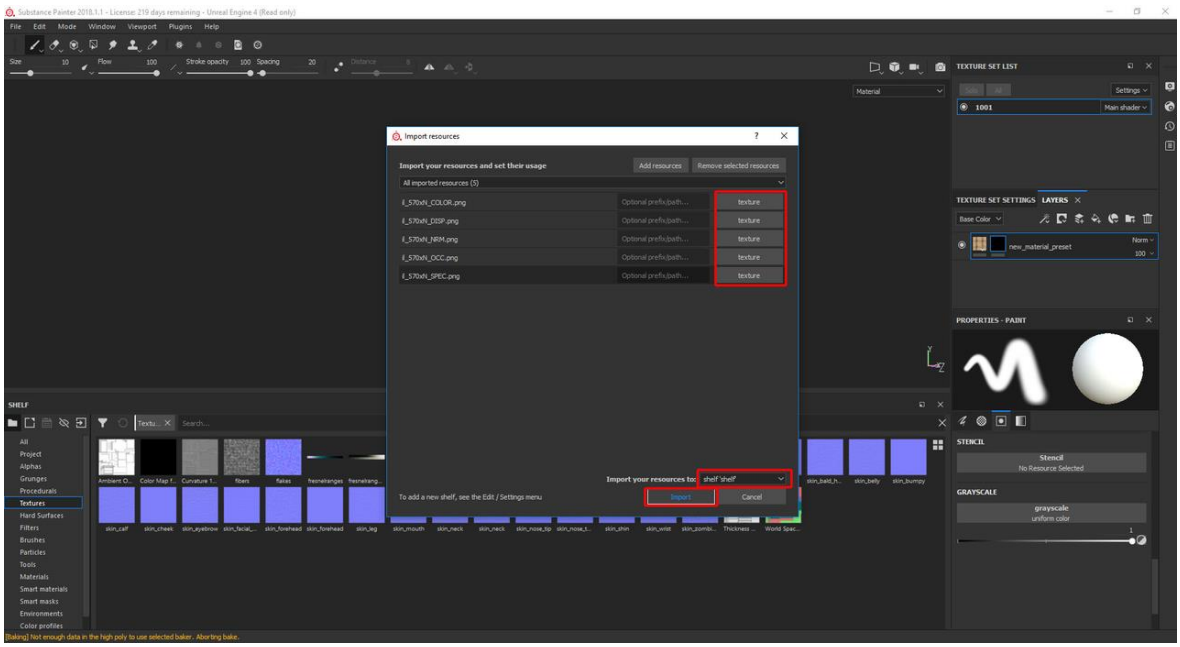


Ilustración 31.Importar mapas a Substance Painter

Y, por último, teniendo seleccionada la herramienta de pincel, a la derecha donde nos aparecen las propiedades de la pintura, arrastramos los mapas que corresponden a cada espacio. Una vez hecho esto, hacemos clic sobre la esfera de previsualización y le damos a "Create material preset" y ya tenemos nuestro material creado para poder ser utilizado.

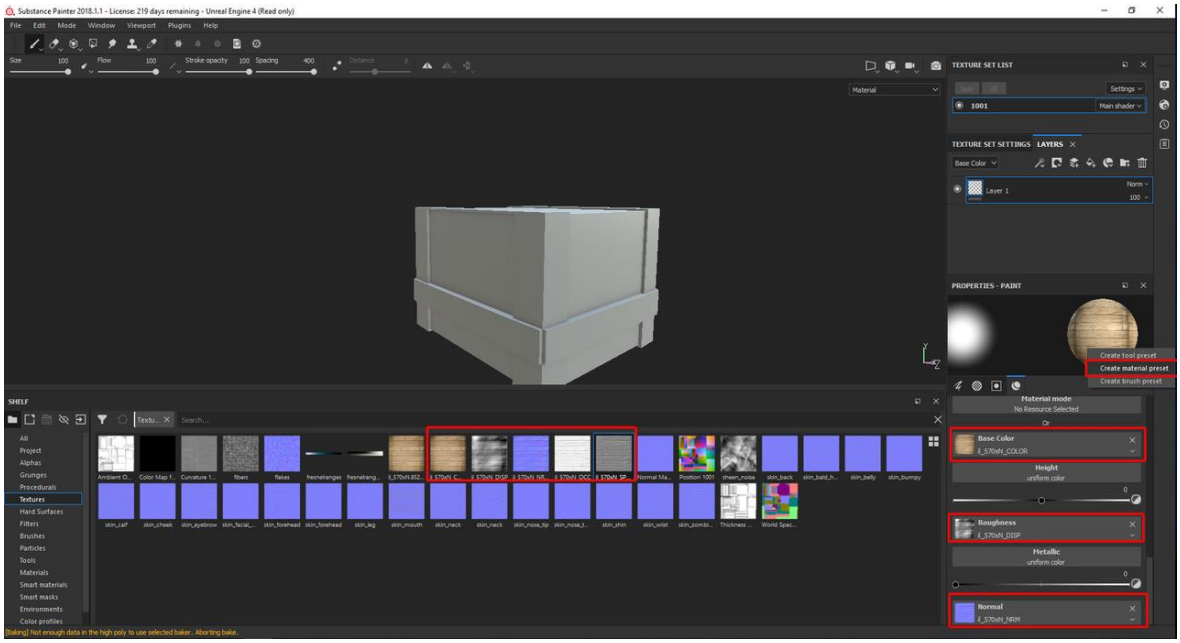


Ilustración 32. Creación de material a partir de los mapas importados.

Substance Painter

Para crear una textura que se adapte bien a nuestro objeto vamos a utilizar este programa. Para ello vamos a importar el modelo en formato ".fbx" y como son texturas para videojuegos, pondremos que tenga una resolución de 1024 para no sobrecargar la escena en UE4.

Lo siguiente que haremos antes de empezar a pintar y a aplicar texturas a nuestro objeto, será crear los mapas del objeto haciendo *backing*.

Backing es el proceso de transferir detalles a un modelo 3D, normalmente se utiliza para crear un mapa de normales y dotar a un modelo Low Poly de detalles que solo un modelo de alta resolución tendría. El *backing* se hace una vez tengamos los siguientes puntos:

- Modelo Low Poly con mapa de UV creadas.
- Modelo High Poly de ese mismo objeto con los detalles que deseemos.
- Un "Cage" que creará una copia "inflada" del modelo Low Poly respecto al High Poly.

Por último, se tienen que poner ambos modelos en la misma posición.

Una vez tenemos lo anterior, nos dirigimos en Substance Painter a la ventana de "Texture set settings" y le damos a la opción "Texture backing" y aparece una ventana en la que añadiremos el modelo High Poly donde elegiremos los parámetros de la "Cage" antes mencionada. Cuando tengamos esto hecho le damos a "Bake all textures" y nuestro modelo Low Poly tendrá un aspecto mucho mejor y muchos más detalles, casi hasta como el modelo High Poly.

Además, cuando hacemos *backing* en Substance Painter también se crea un mapa de posiciones que te permite detectar bordes, lo que esta abajo y arriba, recovecos; y cuando le apliquemos, por ejemplo, filtros de suciedad, estos se distribuirán de manera inteligente por el modelo, por donde realmente la suciedad se acumularía.

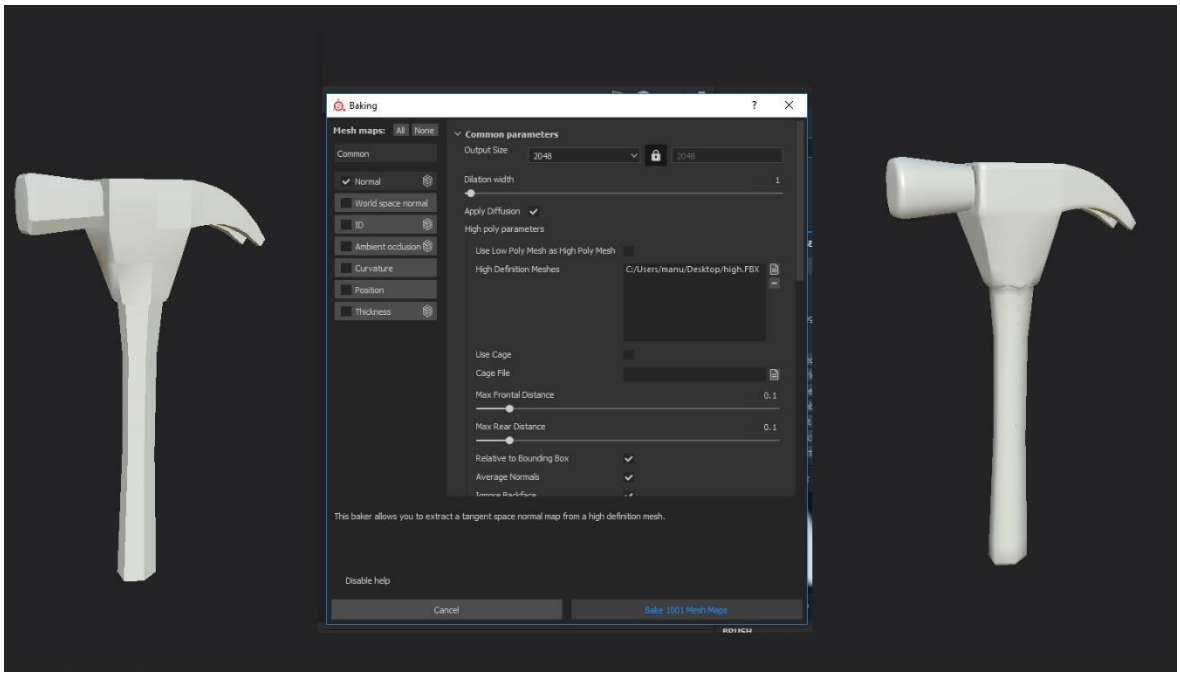


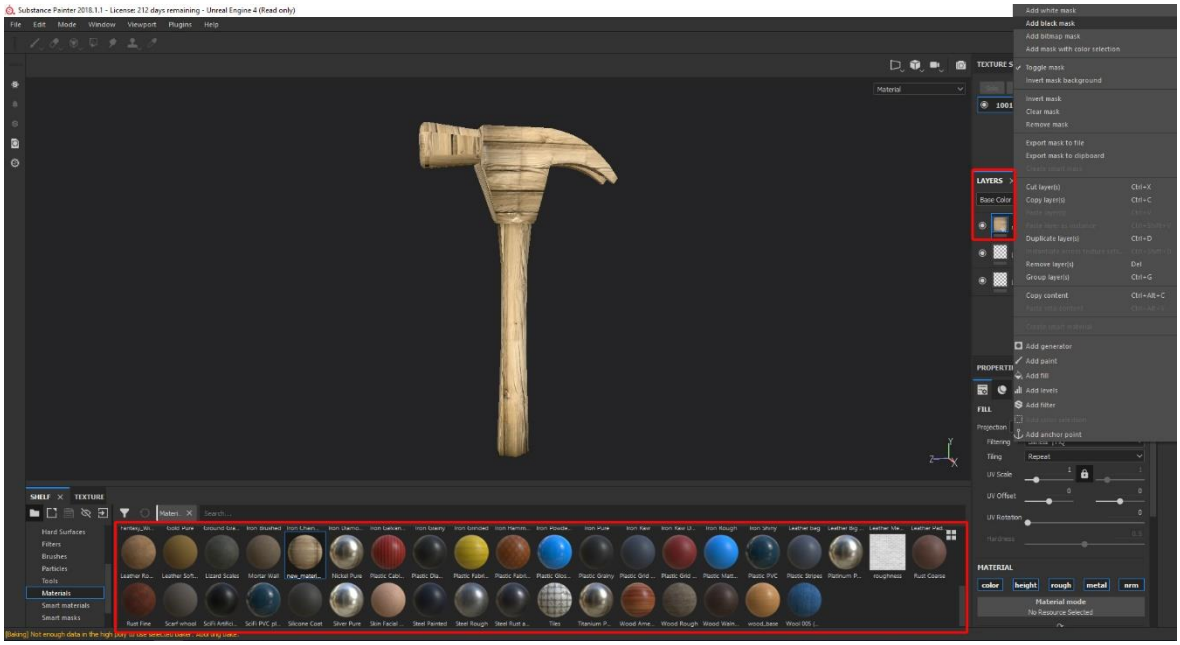
Ilustración 33. Proceso de backing.

A la izquierda vemos cómo en el modelo antes de ser backeado se aprecia el cambio entre polígonos, y a la derecha se ve todo más suavizado sin haber aumentado la cantidad de polígonos.

Ahora podemos empezar a pintar o aplicar materiales a nuestro martillo que se irán aplicando sobre el mapa de UV.

Lo que vamos a hacer va a ser crear al mango del martillo con la textura de madera anteriormente creada, la cabeza del martillo será metálica y por último le añadiremos un poco de suciedad y aspecto deteriorado para que parezca más realista.

Para empezar, elegimos la textura de madera de la lista de materiales y lo arrastramos a la zona de “layers”, donde aparecerán todas las capas que tendrán la textura, y vemos que se aplica a todo el objeto.



Para que solo se le aplique al mango, lo que hago es crear una máscara negra, que lo que hace es que lo que pintemos de color negro cogerá el color del material y lo que esté blanco no.

En la siguiente imagen se ve con las dos capas ya puestas (marcada con un 1) y las máscaras respectivas para cada una (marcada con un 2) y este sería el resultado:

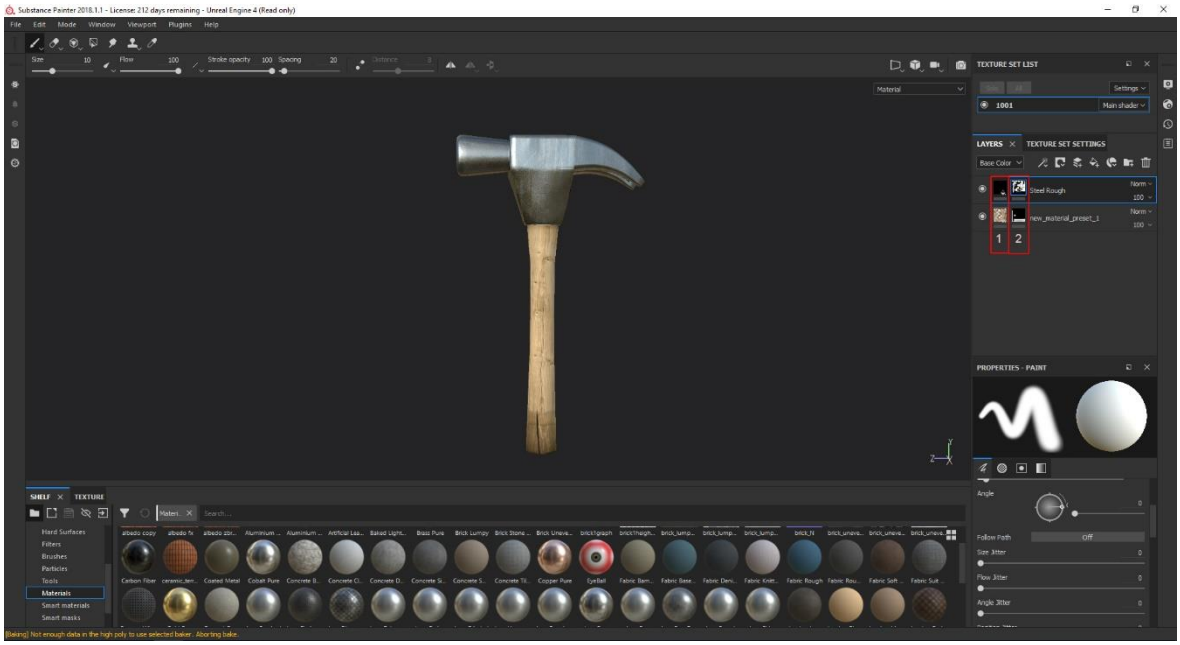


Ilustración 34. Aplicación de materiales mediante máscaras en Substance Painter.

A continuación, mediante las máscaras inteligentes le aplicaremos suciedad y muecas de desgaste que se aplicará a un sitio o a otro en función de cual elijamos. Por ejemplo, si

queremos una de desgaste sabemos que se desgastará más por los bordes debido al roce y demás, entonces automáticamente se verá cómo los bordes se ven afectados por esta. Para crear este efecto tenemos que elegir un color básico, y luego arrastrarle esa mascara que te creará la suciedad o los bordes con ese color elegido, en mi caso he elegido el gris. Para finalizar, en el recuadro de abajo podemos modificar los parámetros de la máscara, como el contraste o la cantidad de suciedad que queramos.

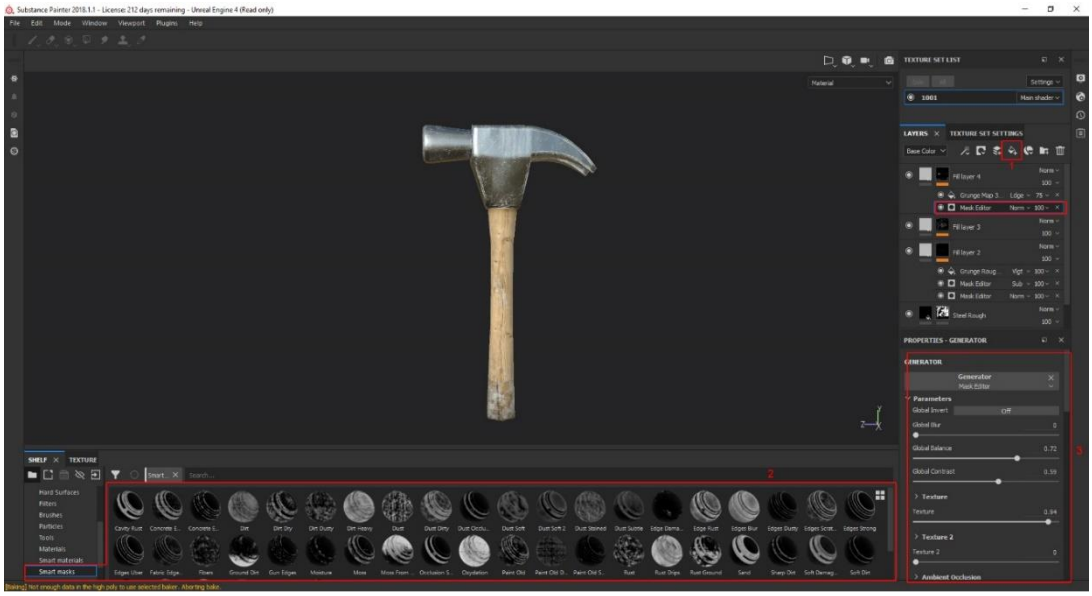


Ilustración 35. Uso de máscaras inteligentes en Substance Painter.

Por último, cabe decir que también podemos utilizar pinceles que añaden detalles al mapa de normales y generar relieves. Por ejemplo, vamos a añadirle un par de tornillos al martillo.

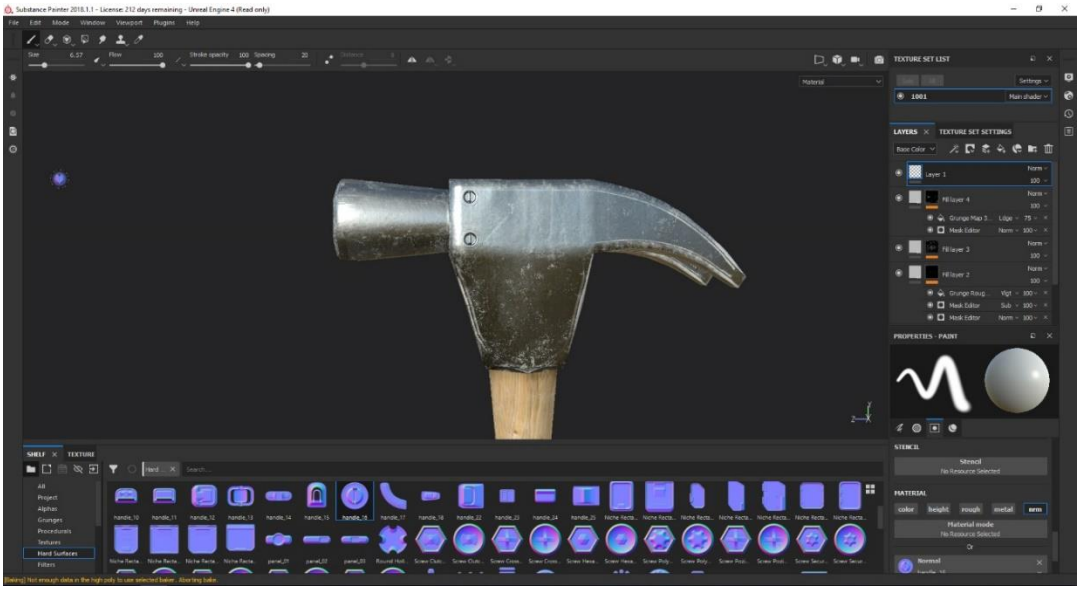


Ilustración 36. Adición de hardsurfaces mediante pinceles con mapas de normales.

Esto ha sido un ejemplo muy básico que permite apreciar lo que hace Substance Painter y he repetido con todos los objetos que hemos modelado.

Exportar

Para poder utilizar el modelo en Unreal Engine, vamos a exportarlo en el formato de Unreal y resolución 1024, pero antes hay que revisar en la pestaña configuración cómo tendremos que utilizar los mapas en Unreal Engine.

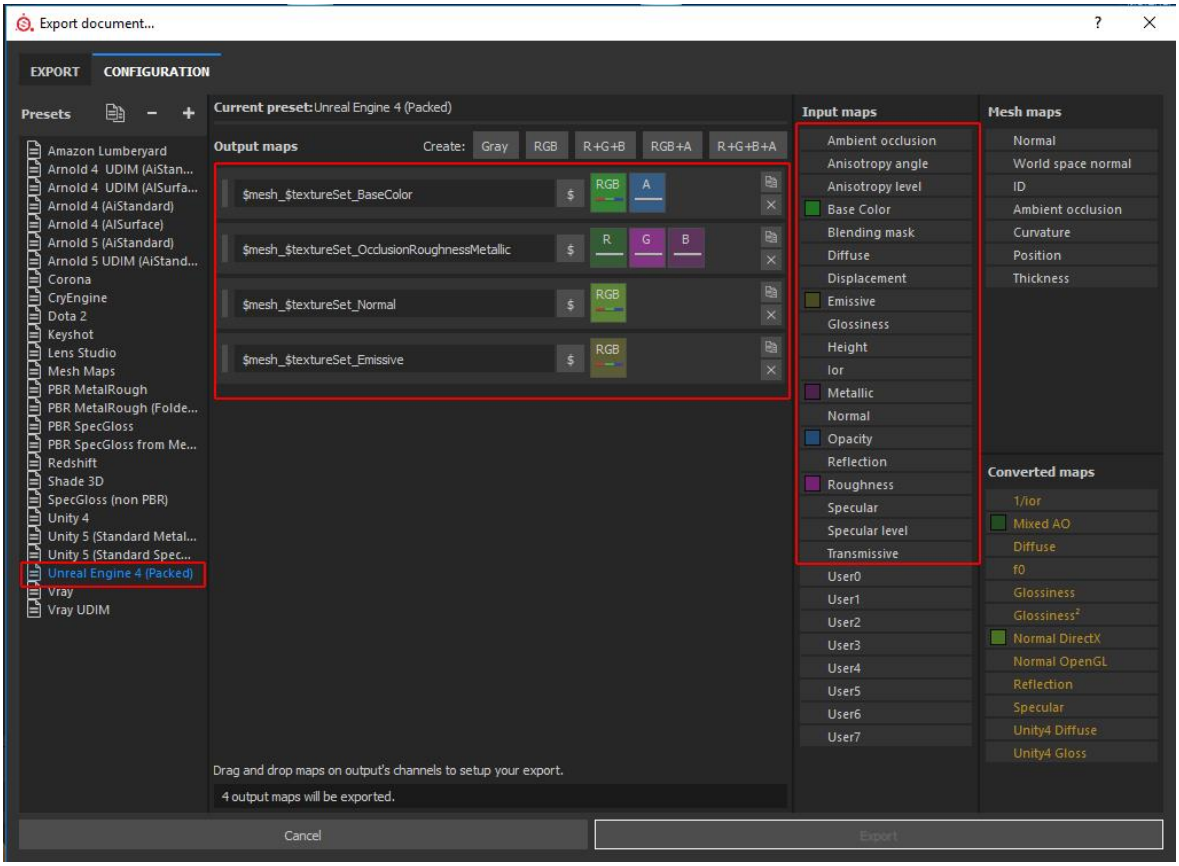


Ilustración 37. División de los mapas de texturas según colores para importar en UE4.

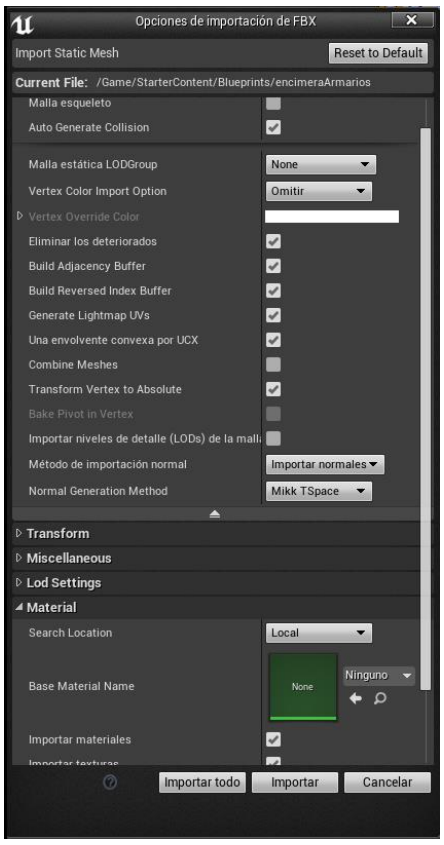
En el recuadro del centro vemos los mapas que vamos a exportar con los diferentes colores en los que se divide el mapa, y en el de la derecha vemos qué color corresponde a cada material en Unreal Engine. Esto lo explico más a fondo en el apartado dedicado solo a Unreal Engine para el editor de materiales.

Resultados Obtenidos

A continuación, muestro unas pocas texturas de las que he realizado como ejemplo para algunos de los objetos que he modelado.

6.3.1.4 Importar assets en Unreal Engine 4

Importar Objetos.



Para utilizar los modelados en formato ".fbx" anteriormente creados en 3dsmax en Unreal, nos dirigimos al buscador donde pone importar y nos aparecerá una ventana parecida a la siguiente.

Dejamos seleccionado lo que venga por defecto a excepción de un par de cosas. Como queremos que nuestro objeto tenga colisión tenemos que asegurarnos de que esté la casilla "Auto generate collision" activada, y segundo tenemos la opción de "Combine Meshes", si tenemos un archivo con un objeto dividido en diferentes partes, deberíamos tener activada esta casilla para que solo cree un objeto, al importarse se crearán todos los huecos de texturas correctamente si se han realizado bien los mapas de UV.

Por último, hay que asegurarse de que tenemos activada la opción de importar materiales y texturas.

Importar Materiales y texturas.

Una vez creadas las texturas en Substance Painter o CrazyBump, las importamos en Unreal Engine 4, y gracias al editor de materiales podremos unir los diferentes mapas de texturas generado en Substance para crear una textura compuesta por todos ellos.

Lo normal es que tengamos 3 mapas de texturas, base color, normal y un solo mapa que contiene el metálico, la desigualdad y el ruido ambiente.

Para importar las texturas lo que hacemos será dirigirnos a "Buscador de Contenido" y darle a importar (no podemos arrastrar a la carpeta del proyecto), y seleccionamos todos los mapas que vayamos a necesitar. Para poder añadir estos mapas a los objetos hay que hacerles algunos ajustes y crear el material que englobe todos esos mapas, que lo explicaremos en el apartado de implementación en UE4.

6.4 Implementación en Unreal Engine 4.

6.4.1 Conceptos básicos

Unreal Engine 4 es un motor gráfico completo con todas las funcionalidades que un juego AAA necesita, por lo que la cantidad de funcionalidades que tiene es inmensa. Por ello, solo voy a hablar sobre las funcionalidades y conceptos que nos pudiesen hacer falta en nuestra escena. [11]

6.4.1.1 Niveles

Cuando juegas a un videojuego, todos los objetos que ves o con los que interactúas, están dentro de lo que se llama Nivel. En UE4, un Nivel está formado por una colección de mallas estáticas, volúmenes, luces, Blueprints y todo lo que genere la experiencia deseada al jugador.

Un videojuego puede tener todos los niveles que desees. Podría tener una pequeña escena con varios niveles, o una inmensa con solo un nivel.

6.4.1.2 Actores

Un actor es cualquier objeto que puede ser añadido al nivel. Los actores son una clase genérica que soporta transformaciones geométricas 3D (rotaciones, translaciones, escalado, etc.). Existen diferentes clases de actores, entre ellos están CameraActor, StaticMeshActor y PlayerStartActor, que se podría decir que son los más básicos. Las luces, los volúmenes y los sistemas de partículas también son actores.

Ilustración 38.Imagen que muestra los diferentes tipos de actores en UE4

Para realizar la escena hay que tener claro cuáles son cada uno de los actores y sus características básicas para que interactúen entre sí de manera correcta para crear un buen nivel y una experiencia satisfactoria.

Cuando añadimos los objetos a la escena hay que tener en cuenta si el objeto va a permanecer siempre en el mismo sitio, porque dependiendo de ello le afectará de una manera u otra las luces y las sombras que proyectará. Las distintas opciones que tenemos respecto a la movilidad son las siguientes.

- **Static**: Esta característica se les da a los actores que no se pretenden mover durante la ejecución del juego.
    - **Mesh Actors:** Una vez tenemos las sombras calculadas y el mapa de luces, estas ya no varían respecto a ese objeto. Esto viene bien para estructuras y elementos de decoración que no se van a mover y consumir menos recursos al ahorrar cálculos de luces y sombras durante toda la ejecución del programa.

- **Light Actors:** Con esta opción se contribuirá a construir los mapas de luces precalculados. Continuará iluminando objetos dinámicos debido a la caché de la luz indirecta.

- **Stationary**: Esto se utiliza solo para las **luces** que no se van a mover, pero que se van a actualizar en algún momento durante la ejecución del juego, como que se enciendan y se apaguen o cambien el color y el brillo, pero no podrán generar sombras en movimiento.

  Algo muy importante en estas luces es el **LightMap Resolution,** que hará que las luces y sombras que se generen tengan mayor o menos calidad.

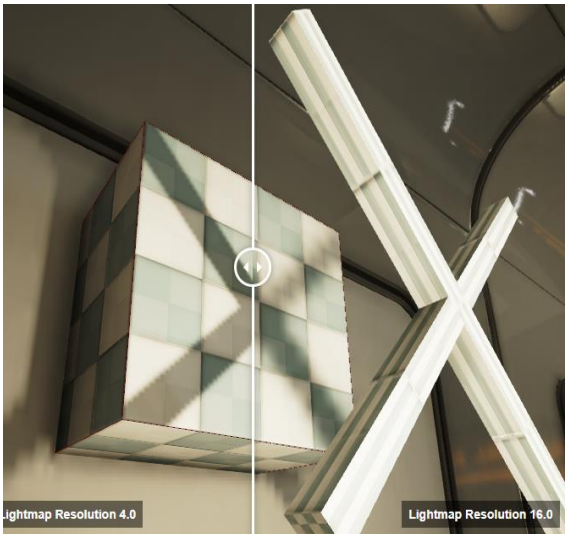
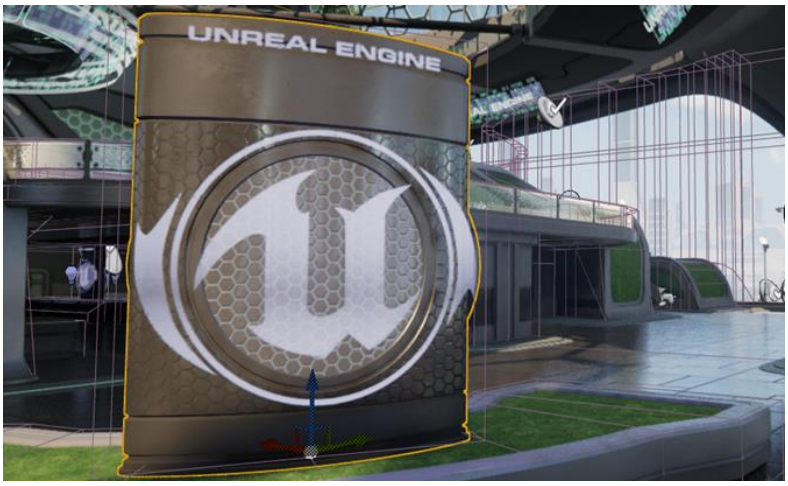


Ilustración 39. Comparación de la resolución del Lightmap.

- **Movable**: Esta opción se usará solo en el caso de actores que se vayan a mover durante la ejecución del juego. El uso de esta opción consume muchos más recursos que las otras dos ya que tiene que recalcular las sombras.
  - **Mesh Actors**: No creará sombras precalculadas en el mapa de luces. Pueden iluminarse con luces estáticas debido a la memoria caché indirecta. Proyectan sombras dinámicas si se iluminan con Stationary lights o Movable lights.
  - **Light Actors**: Solo podrán crear sombras dinámicas. Las sombras que generan son las de mayor rendimiento, así que se usarán solo cuando sea estrictamente necesario.

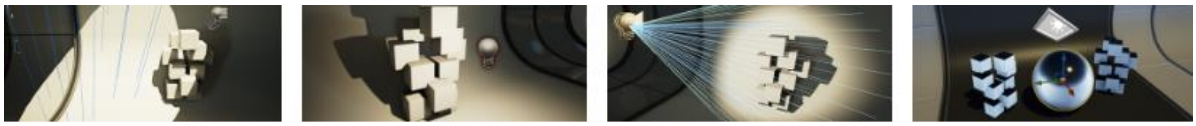
Meshes

Estos actores son básicamente todos los objetos que hemos creado previamente en 3dsmax para que la escena se parezca a la de la película. Más adelante cuando los estemos introduciendo en la escena hablaré un poco más sobre ellos.



Luces

Las luces son los actores que se encargan de iluminar la escena y crear sombras. Existen diferentes 4 tipos de luces básicos en UE4: **Directional**, **Point**, **Spot** and **Sky**.



Directional Lights Point Lights Spot Light Sky Light

- **Directional Light/Luz dirigida:** simula una luz que es emitida desde una distancia muy lejana. Esto significa que todas las sombras que cree esa luz serán paralelas. Esta luz es buena para simular la luz solar.
- **Point Light/Luz con objetivo:** Este tipo de luces actúan como una bombilla, emiten luz en todas direcciones de manera equitativa.
- **Spot Light/Luz de foco:** Emite luz desde un único punto con forma de cono. Esta luz tendrá dos conos para crear la luz, uno interior y otro exterior que hará que se disperse más la luz o se focalice en una zona más pequeña.
- **Sky Light/Luz del cielo:** Esta luz captura las partes lejanas del nivel y las aplica a la escena como una luz. Esto quiere decir que la apariencia del cielo, su iluminación y reflejos coincidirán, incluso si tu cielo viene de la atmósfera o una capa de nubes.

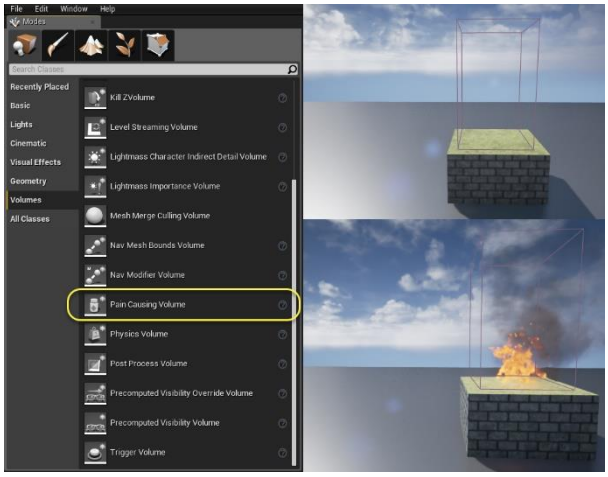
Todas estas luces tienen muchas propiedades que explicaré más adelante conforme las vayamos utilizando y poniendo en la escena.

Sonido y audio

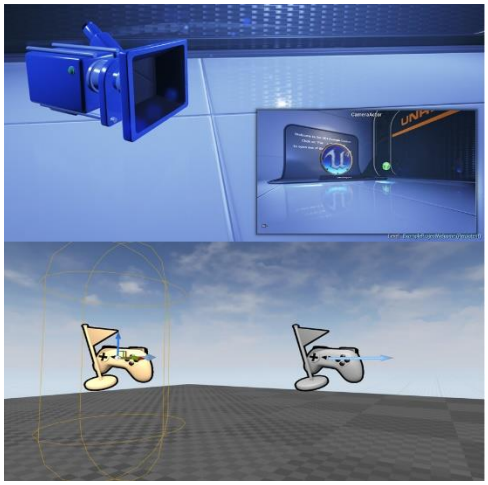
Estos actores permiten dotar a la escena de sonidos y audios para dotar al juego de más realismo. Te permite crear sonidos que se reproducen cuando pasa algo, como disparar, o diálogos entre personajes. También permite asignarles sonidos a los materiales para que cuando, por ejemplo, los pises, suenen dependiendo del material que estén hechos. Asimismo, se pueden añadir sonidos ambientales como tormentas o el sonido de un río.

Volúmenes

Los volúmenes son actores tridimensionales que se utilizan para crear comportamientos en las áreas donde se colocan en los niveles. Los niveles tienen diferentes efectos, como puede ser bloquear a ciertos actores para que no pasen y actuar como una colisión, hacer daño a otro actor, abrir puertas, o activar cualquier funcionalidad que queramos. Por ejemplo, que al entrar en una habitación se encienda una luz.



Camera actor y Player Start.



La manera de visualizar todo lo que creamos a la hora de jugar es mediante el actor Cámara, entonces lo primero que tendremos que crear para poder jugar será este actor junto al Player Start, que hará que el jugador aparezca en el nivel en la localización que le digamos.

Decals

Los Decals son materiales que simulan una especie de pegatinas que se pueden poner sobre la superficie de los objetos de la escena, ocupa un espacio volumétrico, y todos los objetos que estén dentro de ese volumen se verán afectados por este Decal y esa zona del objeto se pintará con esa pegatina. Son de mucha utilidad para crear, por ejemplo, paredes con suciedad, sangre, etc.

Matinee

La herramienta Matinee permite crear animaciones en los actores a lo largo del tiempo, para crear secuencias de juego dinámicas o cinemáticas durante el nivel. El sistema se basa en el uso de pistas de animación en las que se pueden colocar fotogramas clave, lo que permite crear en el punto del tiempo que digamos la posición de los actores en la escena.

6.4.1.3 HUD

El HUD suele ser información representada en 2D que aparece en la pantalla del jugador a modo de interfaz de usuario para marcar ciertos elementos de interés para este, como la vida, la munición, un minimapa o las misiones que tienes pendientes por hacer. El HUD se construye mediante Widgets, que son estos elementos 2D, junto con la ayuda de Blueprints para que se vayan actualizando durante el transcurso de la partida.

6.4.1.4 Blueprints

El sistema de scripting visual mediante Blueprints es un sistema basado en el concepto de usar una interfaz de programación basada en nodos desde el editor de Unreal. Este sistema está basado en un lenguaje orientado a clases u objetos, este sistema es extremadamente flexible y te permite crear absolutamente todo lo que pudiese crear un programador con lenguajes convencionales.

El sistema de Blueprints funciona básicamente conectando nodos, eventos, funciones y variables lo que permite crear todas las funcionalidades del nivel.

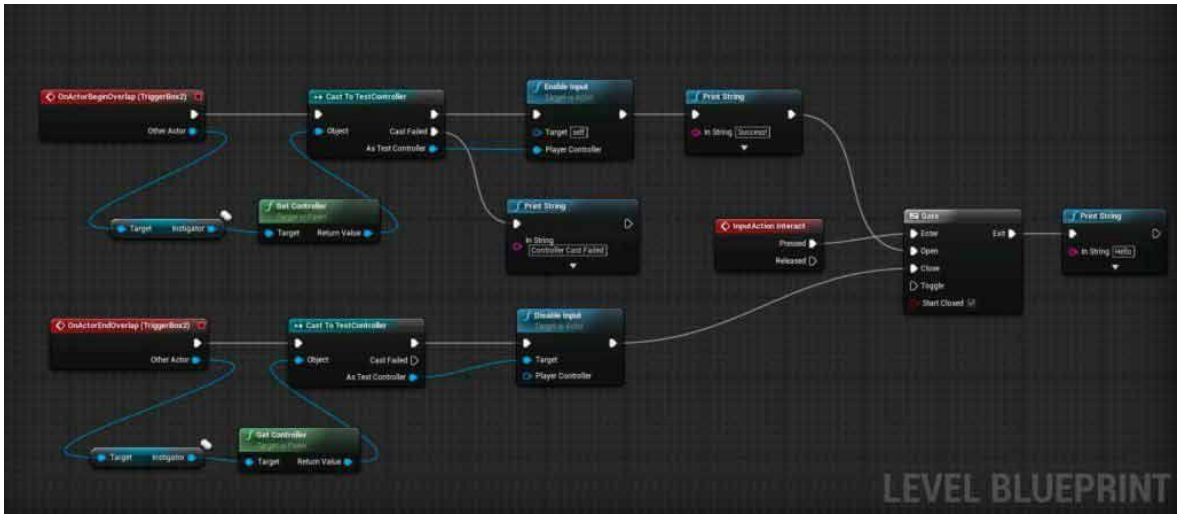


Ilustración 40. Ejemplo de una funcionalidad mediante Blueprints

Tipos de Blueprints

En Unreal Engine 4 existen diferentes tipos de Blueprints. Los más importantes son el "Level" y el "Blueprint clases" que explicaremos a continuación.

Level Blueprints

Cada nivel tiene su propio ¡Level Blueprint. Este puede referenciar y manipular objetos envueltos en el nivel, controlar cinemáticas usando el actor Matinee, crear puntos de guardado o cualquier cosa relacionada con el nivel, como widgets de misión en el HUD.

Blueprint classes

Los Blueprints de clases son perfectos para hacer *assets* interactivos como puertas, objetos coleccionables o escenarios destructibles. Cuando tenemos objetos del mismo tipo y queremos que respondan todos igual ante un evento utilizaremos este tipo de Blueprint.

Elementos importantes

Existen muchos conceptos importantes sobre los Blueprints pero debido a su gran extensión solo hablaré de los que han hecho falta durante el desarrollo del proyecto.

Event Graph

El grafo de eventos de un Blueprint contiene un grafo tipo nodo que usa llamadas a eventos y funciones para llevar a cabo funciones en respuesta a eventos del juego asociados a Blueprints. Esto se usa para añadir funcionalidades que son comunes a todas las instancias

de un Blueprint y es donde todas las respuestas dinámicas e interacciones se ponen en marcha. Por ejemplo, si queremos abrir una puerta presionando un botón, el Event Graph proporcionará este comportamiento a todas las instancias de la puerta con este Blueprint.

Funciones

Las funciones son nodos de los grafos pertenecientes a un Blueprint en particular que puede ser ejecutado o llamado desde otro grafo a través del Blueprint. Las funciones tienen una única entrada designada por un nodo con el mismo nombre de la función que contiene un solo pin de ejecución de salida. Cuando una función es llamada desde otro grafo, el pin de salida se activa y hace que empiece a ejecutarse toda la red que conforma la función con el resto de los elementos.

Variables

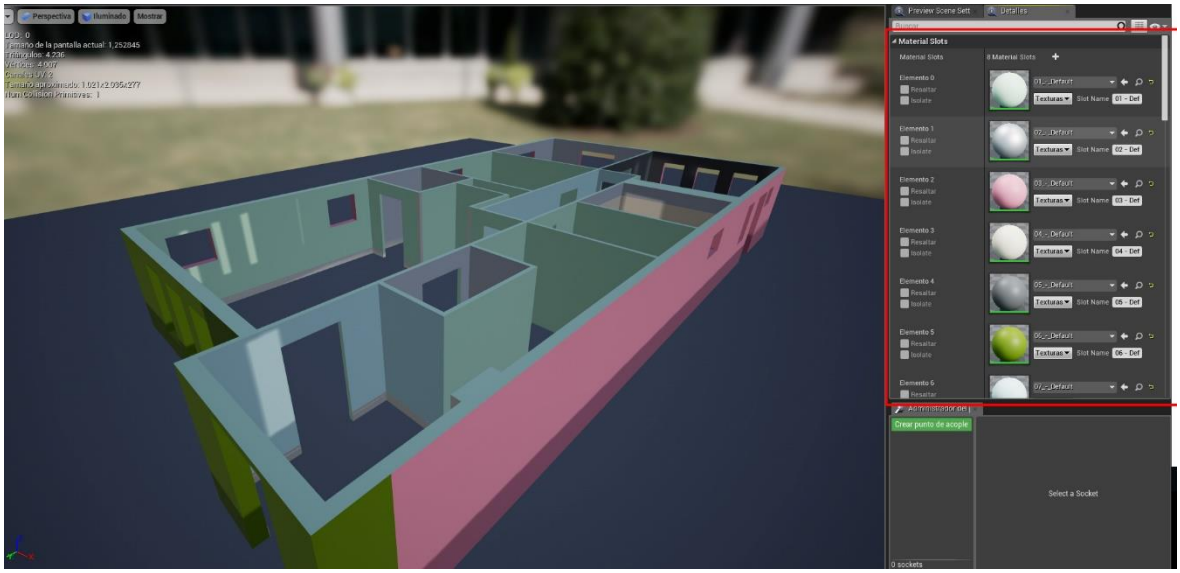
Las variables son propiedades que contienen un valor o referencia a un objeto o actor en el mundo. A estas propiedades pueden acceder internamente los Blueprints que las contienen, o pueden ser accesibles externamente para que sus valores puedan ser modificados por diseñadores que trabajan con instancias del Blueprint que se ha puesto en el nivel.

6.4.2 Creando la escena

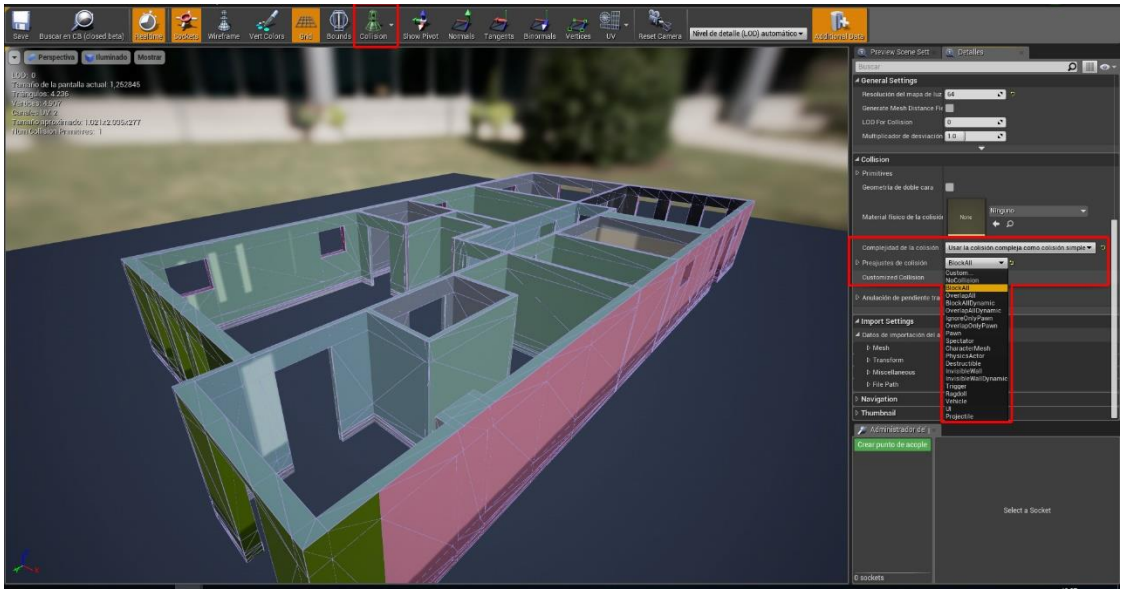
En este apartado voy a explicar cómo he preparado y colocado los *assets* para que aparezcan en la escena y poder interactuar con ellos, cómo he creado la ambientación y la iluminación y, por último, las funcionalidades que se han creado mediante Blueprints.

6.4.2.1 Modelos 3D

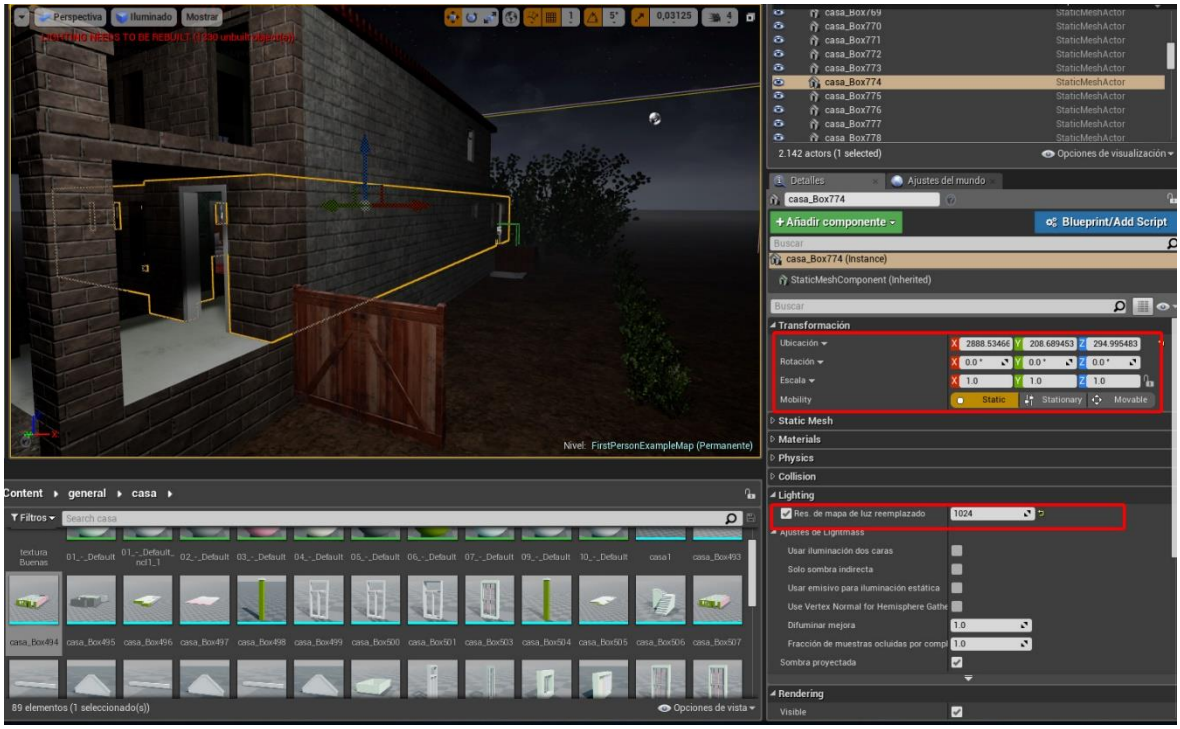
Una vez importados los objetos, hay que hacer ciertos ajustes en ellos para que todo funcione correctamente. Lo primero que vamos a hacer es asignarle los materiales que va a tener. Para ello, si hemos creado la malla correctamente y las UV, cuando accedemos a las opciones del objeto clicando dos veces en él, aparecerá a la derecha un menú de opciones donde se encuentran los diferentes huecos para materiales que se le pueden asignar al objeto.



A continuación, tenemos que crear colisiones para que nuestro actor no pueda ir traspasando las cosas. Hay objetos a los que no hará falta que le hagamos las colisiones, porque al ser objetos pequeños como cuadros o las lámparas de los pasillos, solo estorbaría. Para crear las colisiones, iremos al menú superior donde aparece la opción "colisión". Si le damos a la primera "Simple collision" simplemente crea una caja alrededor del objeto; en muchos casos será más que suficiente, pero si hiciéramos esto con el ejemplo de la casa no se podría entrar a ella, así que vamos a darle a "Complex collision" y se crea una colisión que se adapta a toda la malla (son las líneas azules que se crean por todo el objeto). Por último, pondremos que la complejidad de la colisione sea "usar la colisión compleja como simple" y diremos qué actores se chocarán con esa colisión y cuáles no; en este caso pondremos que colisionen todos.



Por último, una vez colocado el objeto, el cual se puede manipular con W(mover), E(escalar), R(rotar), al seleccionarlo podremos modificar la manera en que le afecta la luz al objeto en la escena con el menú de la derecha. Como ya hemos visto anteriormente, un objeto puede ser “Static”, “Stationary” y “Movable”. Excepto para las puertas que hemos puesto “Movables” para que generen sombras cuando se abran y se cierren, en todas las demás se ha puesto “Static” por motivos de optimización; activamos la Checkbox “Res. de mapa de luz reemplazado”, y le ponemos 1024 a los objetos más visibles para que genere las luces y las sombras con más resolución.



Ahora queda repetir el proceso con todos los objetos de la escena e ir colocándolos según la disposición de estos en la película.

6.4.2.2 Materiales y Decals

Para poder asignarle a los modelados las texturas y materiales que hemos creado e importado a Unreal Engine primero hay que tratarlos. Unreal tiene un sistema de creación de materiales mediante nodos, que permite conectar todos los mapas que habíamos creado y formar un material.

En el editor de materiales podríamos decir que se distinguen tres tipos de elementos, el material con las entradas, los mapas de texturas con las salidas y operaciones. Una vez dentro del editor de materiales, lo primero que nos aparece, es el material con las entradas, lo que tenemos que hacer a continuación es arrastrar a ese espacio de trabajo las texturas que hemos creado y conectarlas donde corresponden.

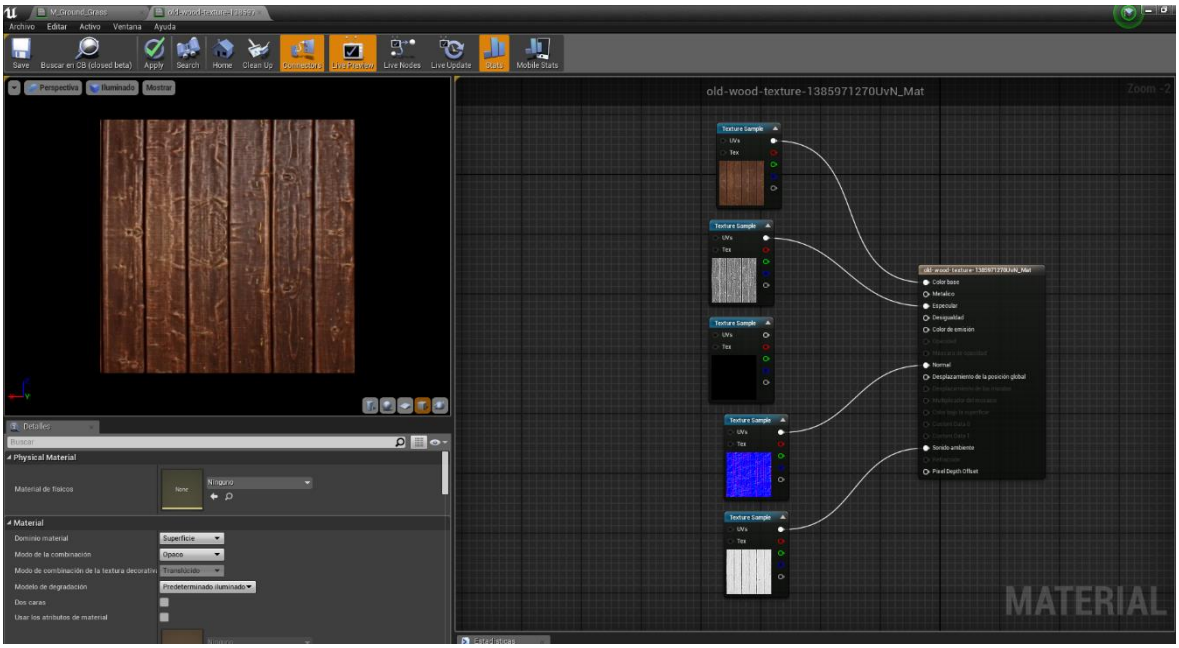
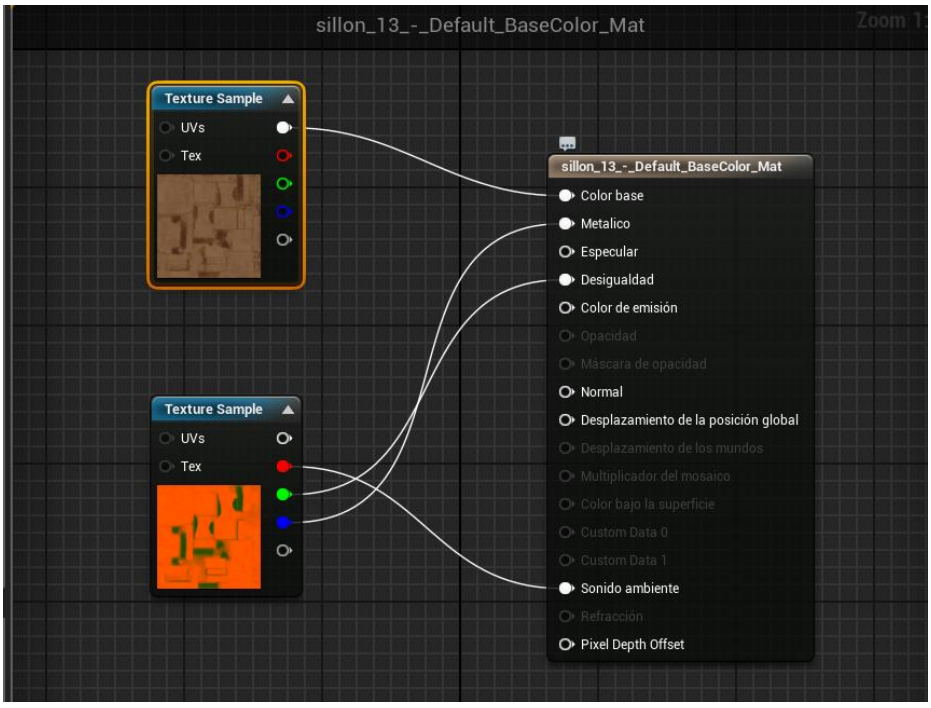


Ilustración 41. Editor de materiales en UE4

En la imagen de arriba se puede observar cómo se ha creado un material que simula madera con su relieve y sus brillos, uniendo las salidas de las texturas con el material. Esto sería lo más básico a la hora de crear materiales. Las texturas tienen diferentes salidas: la primera, que corresponde a todos los colores que conforman la textura; las tres siguientes correspondientes a los colores RGB, que solo utilizan ese color específico para algo en concreto; y la última salida sería el alfa, que la explicaré cuando lleguemos a la creación de Decals. En la siguiente imagen podemos ver un ejemplo en el que los colores rojos corresponden a las zonas donde hay oclusión ambiental, los azules a zonas metálicas y el verde a la desigualdad. Esto último genera brillos dependiendo de cómo de lisa o rugosa sea la superficie en función de, en este caso, los colores verdes; cuanto más oscuro sea el verde más brillos hará.



Como ya hemos dicho, existe un tercer elemento que serían los operadores; los más utilizados son “TextCoord”, “Multiply”, “Lerp”; y nos ayudaremos con parámetros de 1 y 3 valores.

TextCoord: Se utiliza para modificar las coordenadas de una textura o decirle cómo de grande quieres que sea para que se repita más o menos veces una textura procedural. A esto se le llama *tiling*.

Multiply: Se utiliza para multiplicar dos elementos. Se suele utilizar, por ejemplo, para multiplicar una textura por un valor y obtener la salida. Imaginemos que tenemos la textura que define las zonas metálicas pero esas zonas tienen unos niveles muy altos de efecto metálico y necesitamos reducirlo, pues este sería el método.

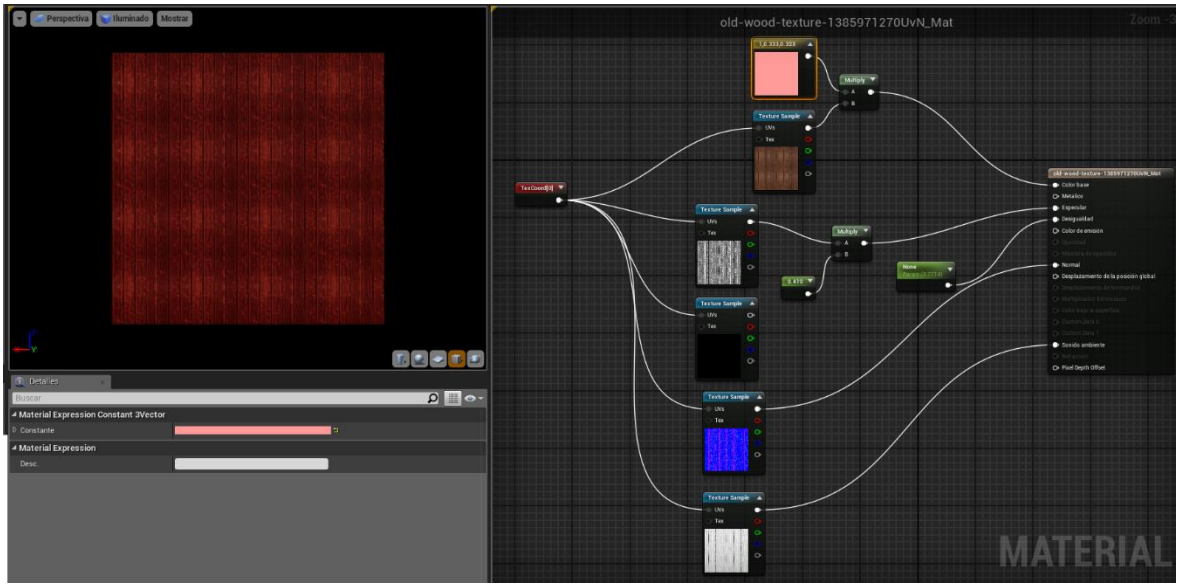
Lerp: Este operador hace una interpolación lineal. Se suele utilizar para realizar máscaras con las alfas de la textura.

Parámetro de 3 valores: Este define un color RGB. Se suele utilizar para conectarlo al color base, o para multiplicar la textura con el color base para modificar el color de la textura.

Parámetro de 1 valor: Esto es una simple variable de un valor. Se puede multiplicar por una textura y modificar su intensidad de metálicos, especular, desigualdad, etc., o también se puede conectar directamente a las entradas del material y definir todos esos valores desde

cero. Por ejemplo, si conectamos una variable con valor 0 a la entrada especular, el material no generará brillos, pero cuanto más nos acerquemos al valor 1 más brillos generará.

Veamos con el ejemplo de antes de la madera cómo se han utilizado estos operadores. Se ha utilizado el operador **TextCoord** para aumentar la cantidad de veces que se repite la textura, el parámetro de 3 valores junto al **Multiply** para cambiar la textura de marrón a rojo, y con los parámetros de1 valor se ha modificado la intensidad de la textura especular y la desigualdad.



Una vez tenemos todas las texturas creadas, se las asignamos a los objetos y tendríamos unos resultados como el siguiente:

Ilustración 42. Imagen de la escena con las texturas aplicadas.

En la imagen de arriba ya estarían puestas las texturas, pero está todo muy limpio, así que lo que haremos será ensuciarlo un poco para darle un aspecto más deteriorado y más terrorífico. Para ello haremos uso de los Decals**,** para poder utilizar estas "pegatinas" que pondremos por toda la casa necesitaremos crear un material con unos valores un poco diferentes y necesitaremos hacer uso de las alfas. Para crear un Decal tendremos que decirle en los parámetros de este que sea un material de tipo "Textura decorativa diferida", modo de combinación "Translúcida" y en el modo de combinación de la textura diferida pondremos "Desigualdad, normal, color translucido del DBuffer". Al hacer esto en el material se activará la entrada opacidad.

Lo siguiente que haremos será arrastrar la imagen que queramos en el editor y asignarle el color al color base y el alfa a la opacidad. Si tenemos un PNG basta con asignarle la última salida de la textura que representa el color alfa a la opacidad, si por el contrario carece de alfas habrá que crear una máscara en Photoshop con colores blancos y negros, donde los negros serán transparentes y los blancos los opacos.

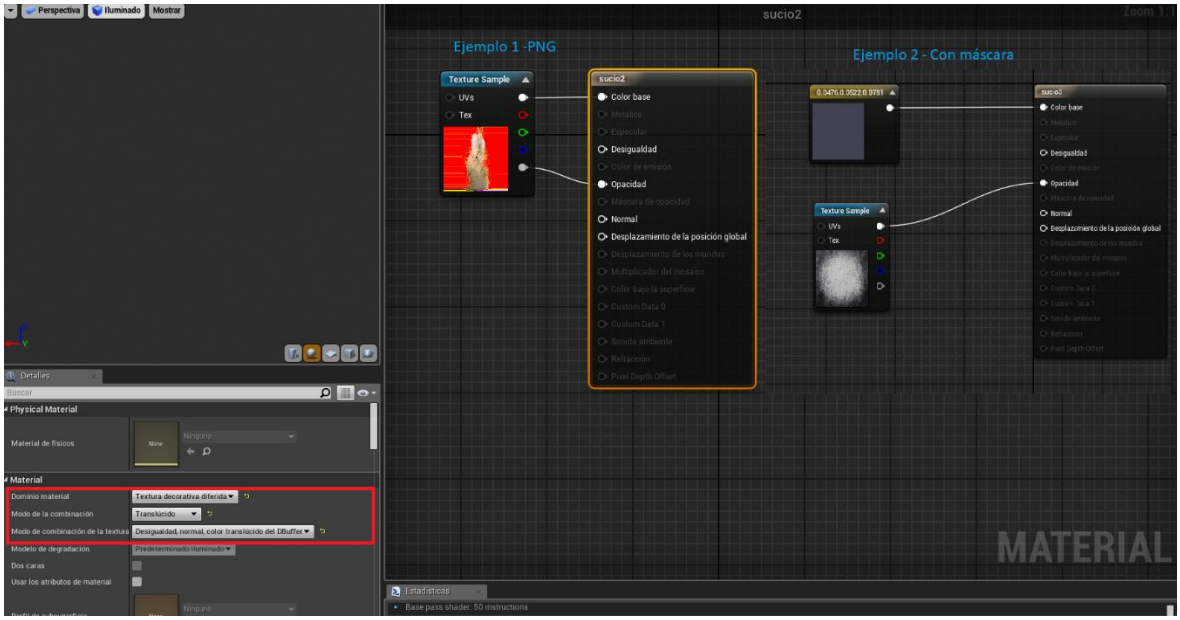


Ilustración 43. Ejemplo de creación de un material tipo Decal.

Una vez tenemos el material lo único que hay que hacer es arrastrarlo a la imagen y aparecerá el volumen que contiene la pegatina y todo lo que se encuentre dentro de dicho volumen, se pintará. Una vez tenemos puesto los Decals conseguimos el siguiente resultado.

Ilustración 44. Escena con los Decals aplicados

6.4.2.3 Luces y efectos ambientales.

Iluminación del interior de la casa

La iluminación de la casa, como en la película, tiene una iluminación tenue, generada por lamparillas y la poca luz que entra del exterior de la casa. Para generar este tipo de iluminación hemos usado solamente dos tipos de luces, Point y Spot lights. La primera se ha utilizado para las lámparas que se encuentran por la casa y, a veces, como apoyo con intensidades muy bajas para ciertas esquinas donde no llegaba apenas luz. Por el contrario, la segunda se ha utilizado como luz general, para iluminar espacios grandes.

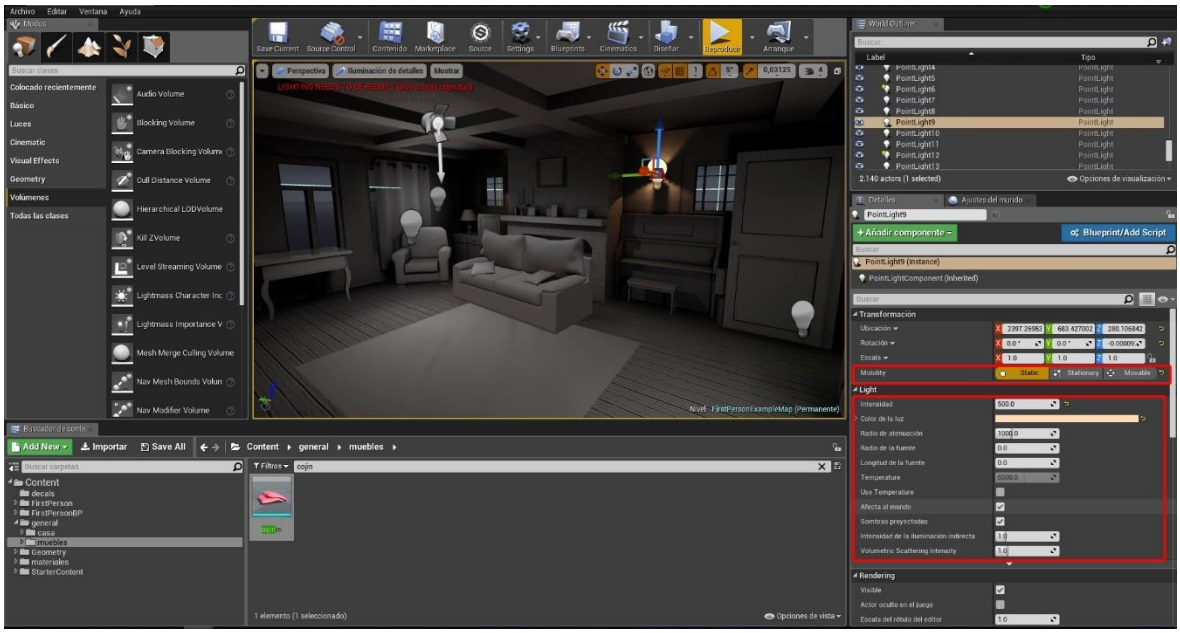
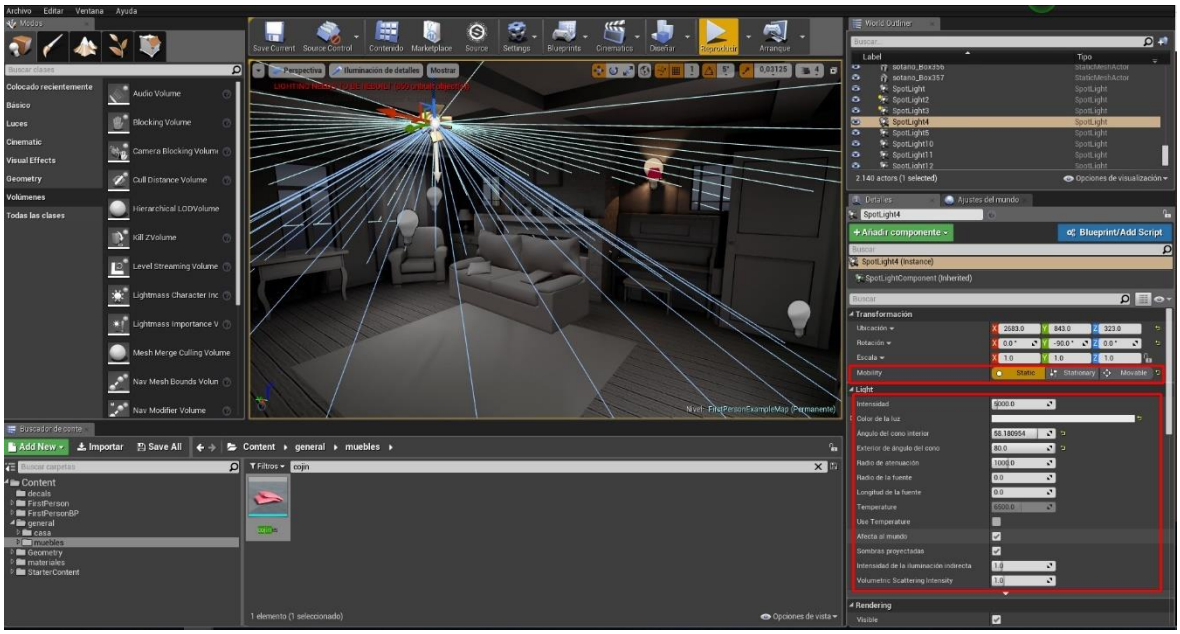


Ilustración 45.Visualizacion de la escena para construir las luces.

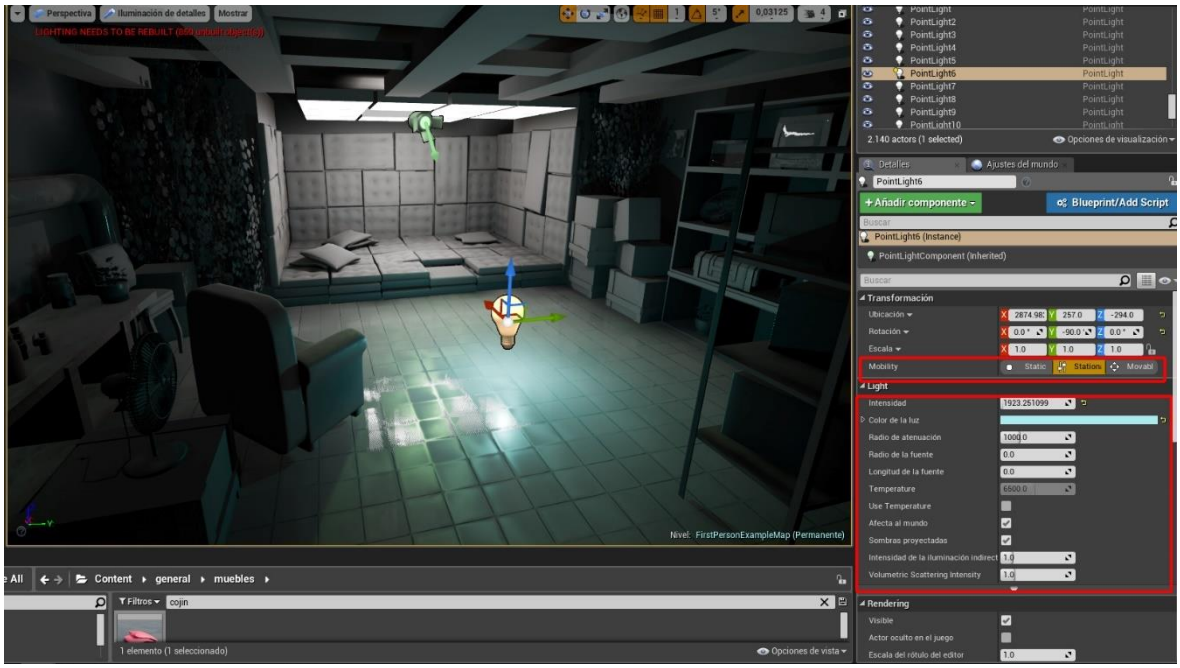
En esta primera imagen vemos las opciones que nos permite la Point light; la hemos colocado dentro de las lamparillas para que cree ese efecto. En los valores de la derecha se ha puesto que sea una luz estática para optimizar la escena, ya que como hemos visto anteriormente, las luces estáticas no tienen que ir actualizando el Lightmap e ir creando las sombras en tiempo real. Además, le he puesto colores más cálidos que el resto, un color anaranjado característico de las lamparillas e intensidades bastante bajas y atenuaciones bastante definidas para crear ese umbral anaranjado que se aprecia en la imagen.

Aparte de las Point lights de las lamparillas, he colocado en puntos que quedaban muy oscuros algunas luces más de este tipo, con colores más fríos, intensidades muy bajas y

atenuaciones mucho menos marcadas para no quemar ni generar brillos en los objetos que estaban cercanos a las luces.



La iluminación con la Spot light, de la figura de arriba, la utilizamos para hacer una iluminación general de toda la habitación porque se quedaba muy oscura. Como esta luz tampoco se tenía que actualizar, se ha puesto de tipo estática y la intensidad se ha aumentado bastante, el cono exterior se ha aumentado al máximo para que englobase toda la habitación y el cono interior se ha dejado intermedio para que no hubiese un corte repentino de luz en las paredes, y así se conseguía una atenuación más que aceptable.



Por último, arriba he puesto un ejemplo de la iluminación del sótano, y aquí como se puede ver en el desplegable de opciones, sí están con una luz estacionaria. Esto se debe a que esta luz sí que la vamos a actualizar durante la ejecución del juego mediante un tipo de material diferente, que hará que la luz parpadee. Además, también se lo asignaremos a los objetos que queremos que emitan luz.

Para crear este material tenemos que dirigirnos al editor de materiales y simplemente poner en "Dominio de luz" que tenga funcionalidad de luz, y así se queda activada la opción de color de Emisión en las entradas del material. Para hacer el efecto de parpadeo, debemos usar el operador "Time" y "Sine" que harán que cada cierto tiempo se apague y se encienda la luz, y gracias al "Frec" lo que hacemos es que haga un parpadeo aleatorio al que le podremos aumentar la frecuencia, para así generar el típico parpadeo de las luces led que aparecen en videojuegos y películas de terror. En el caso de que no se ilumine el material y no emita el color que debería es porque en el menú opciones del proyecto tenemos que activar la opción "Bloom".

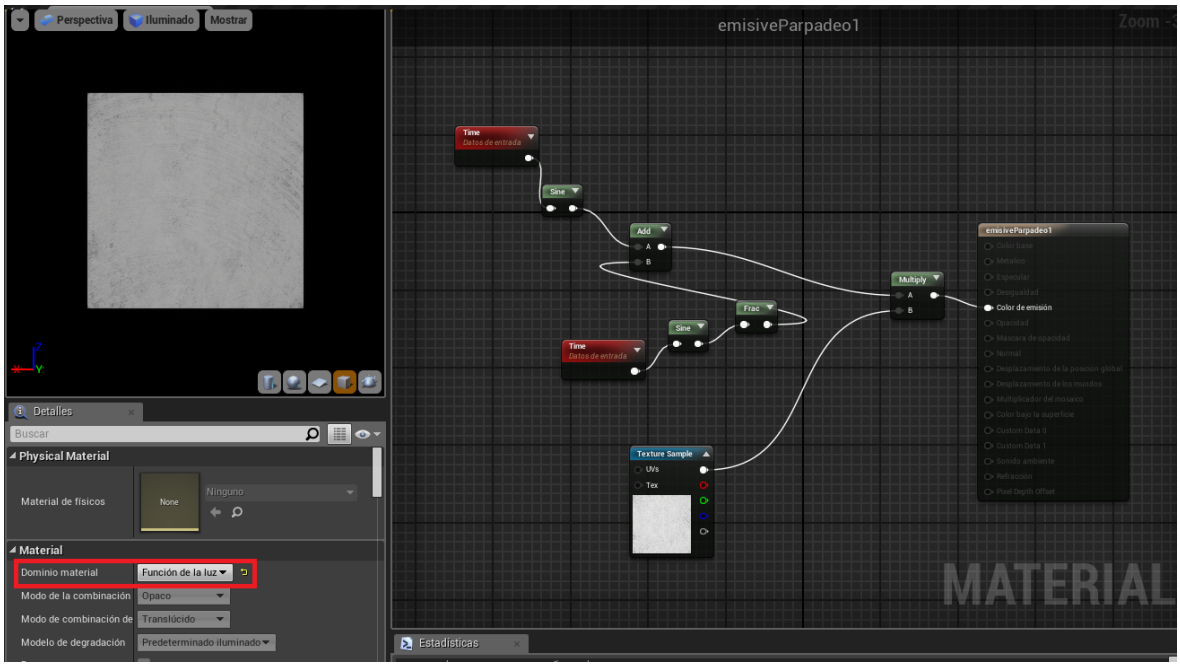


Ilustración 46. Material de tipo luz con animación de parpadeo.

Una vez hecho esto, tenemos que asignarle el material al objeto que queramos y empezará a iluminarse y parpadear, y también se lo añadiremos a las luces asignándole el material a la opción de "Comportamiento de la luz", que solo te dejará asignarlo si es una luz de tipo "Static" o "Movable".

Efecto noche

En la película, cuando entran a robar en la casa es de noche, así que mediante el Blueprint "SkySphere" haremos que desaparezca el sol y el cielo azul y se transforme en una noche estrellada cambiando los parámetros de este.

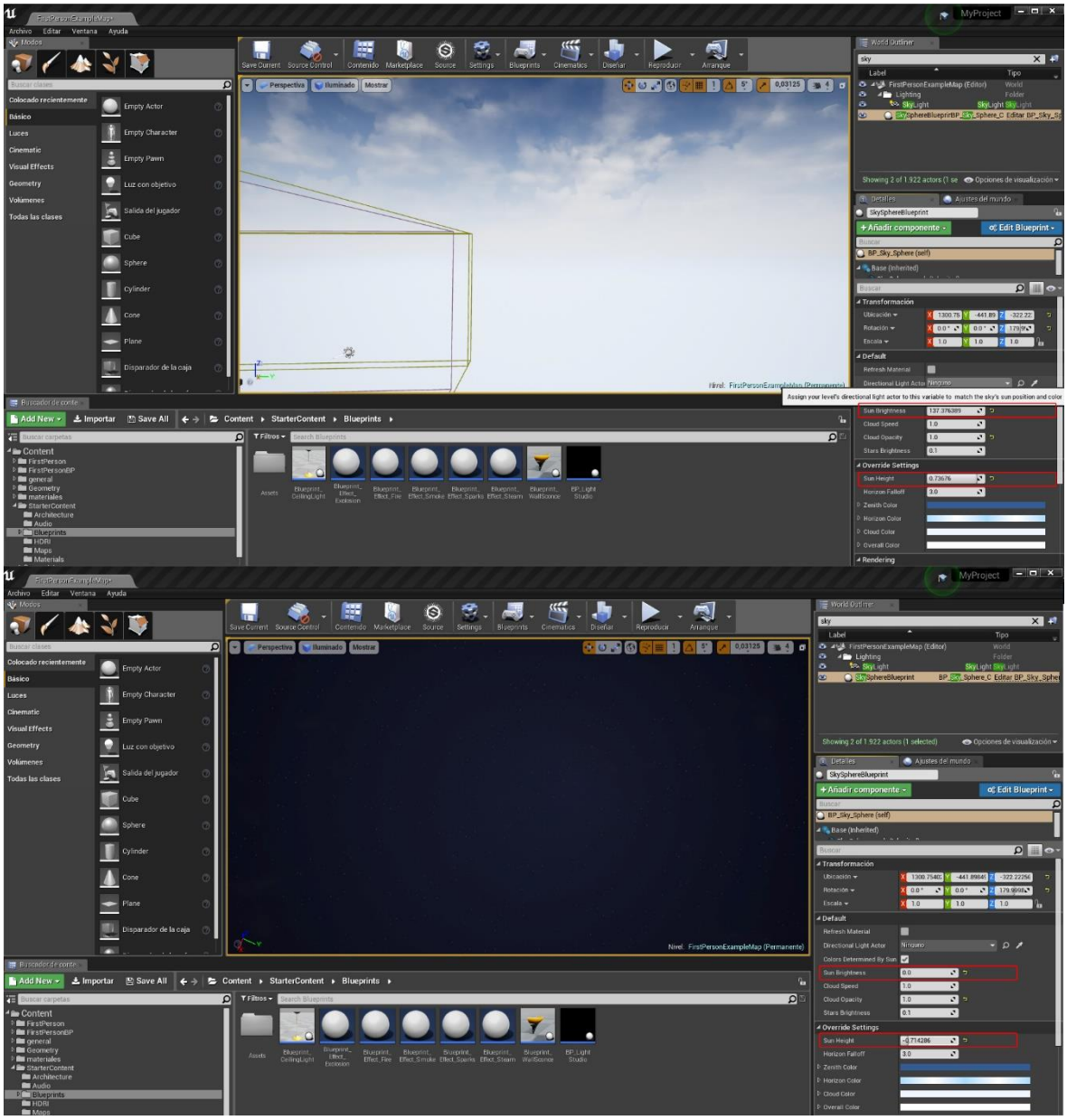


Ilustración 47. Transformación del cielo a modo noche mediante el Blueprint SkySphere.

Lo más importantes aquí es el **"**sun brightness", que al cambiar el parámetro a 0, el dibujo del sol desaparece completamente; y el segundo parámetro "sun height" hará que cambie el

color del cielo, cuanto más bajemos este parámetro más oscuro se irá haciendo, pasando por el mediodía y el atardecer.

El resto de los parámetros son para el gradiente de color entre el horizonte y el cielo o para modificar la opacidad y velocidad de las nubes.

A continuación, vamos a bajarle la luz porque a pesar de que hemos cambiado el cielo a oscuro, la luz sigue siendo la simulada por el sol. Esto lo cambiaremos con la "luz direccional". Lo que haremos será bajar la intensidad y cambiarle el color de la luz a uno más oscuro.

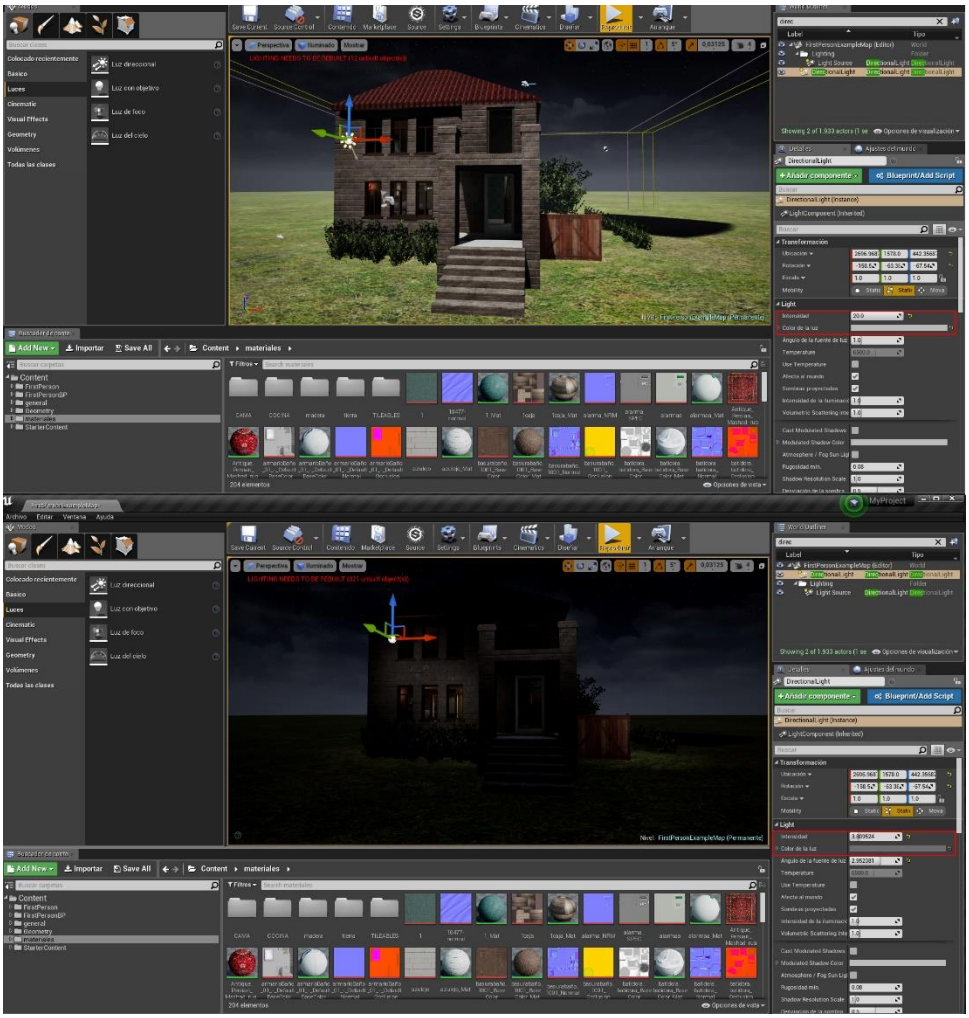


Ilustración 48.Creción efecto noche mediante luz direccional.

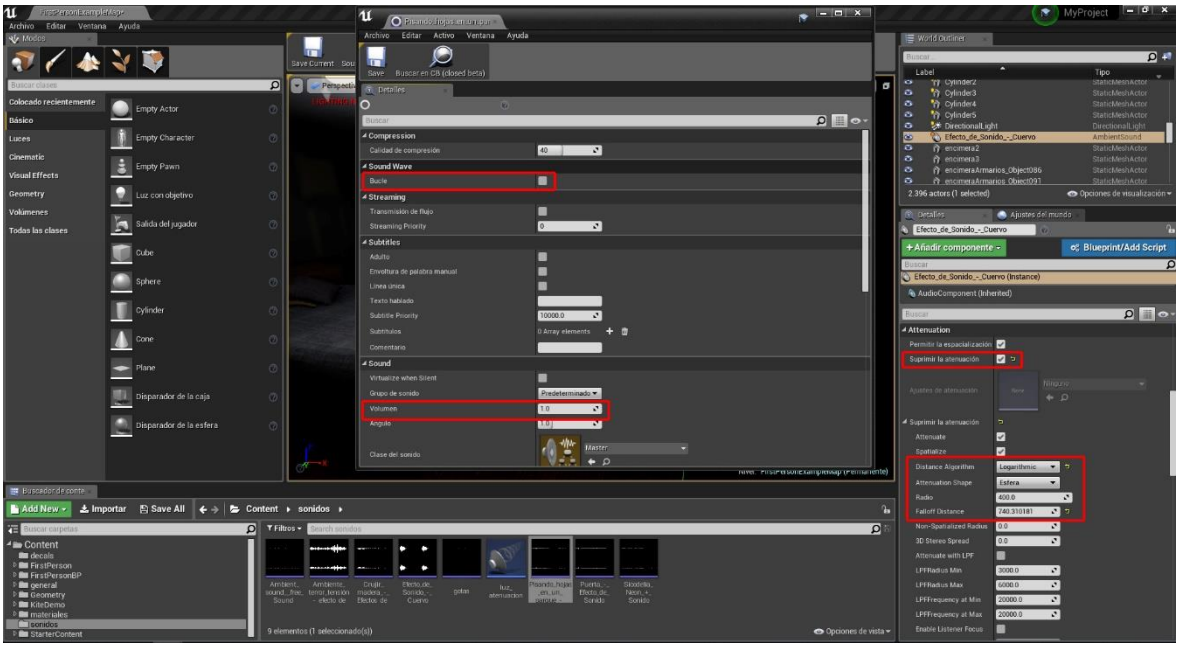
Y por último para darle un efecto más terrorífico le pondremos niebla, que está en la sección "Visual effects" y se llama "niebla atmosférica". Los parámetros para modificarla son muy intuitivos y te permite que haya más o menos densidad y a cuánta distancia se puede percibir, entre otros parámetros.

6.4.2.4 Sonidos

Los sonidos son una de las partes más importantes para dar vida a un juego, nos permite poner en tensión y dar sustos para así conseguir una mayor inmersión del jugador.

Para crear los sonidos en UE4 necesitamos archivos de audio que los podemos conseguir de repositorios o canales de Youtube que nos los facilitan de manera gratuita. Una vez tenemos los audios que queremos, los podemos editar con Audacity de manera muy sencilla para dejarlos a nuestro gusto. Una vez los hemos editado tenemos que asegurarnos de exportarlos en un formato de audio ".wav" para que UE4 los importe bien.

Cuando los tenemos importados lo único que hay que hacer para que se escuchen es arrastrar el sonido a la escena y dependiendo de lo que queramos conseguir modificaremos ciertos parámetros. En la siguiente imagen he marcado en rojo los parámetros que nos harán falta:



Al hacer doble clic en el sonido aparecerá una ventana central con los parámetros del audio importado. Aquí, dependiendo de lo que queramos conseguir, nos puede interesar que el sonido se reproduzca en bucle todo el tiempo o aumentar el volumen del audio, a parte de otros muchos parámetros que a nosotros no nos va a hacer falta.

Una vez metemos el audio a la escena, clicando en el actor nos saldrán en la ventana de detalles todos los parámetros del actor y aquí lo más importante y lo que nosotros vamos a utilizar será el efecto de atenuación del sonido. Para poder ponerle al sonido una atenuación dependiendo de la distancia a la que nos encontremos tenemos que activar la casilla “Suprimir la atenuación” y ya nos dejará modificar los parámetros que necesitamos. Los parámetros que vamos a modificar serán los del recuadro rojo y pondremos en “Distance algorithm” la opción “Logarítmica”. Esto lo que hará es que se escuchará el sonido dependiendo de adonde estemos mirando y así sonará más fuerte o más flojo y por qué auricular saldrá el audio, es decir, si hay un sonido que está ubicado a nuestra izquierda se escuchara por ese auricular y si está en frente lo escucharemos por los dos. Otros parámetros muy importantes son el “Radio” y el “Fallof distance”. El primero será la zona donde sonará más fuerte el sonido y fuera de esa zona el sonido empezará a sonar cada vez más flojo hasta extinguirse, que es el segundo parámetro.

Si queremos que sea un sonido ambiente que suene todo el tiempo y desde el principio de la ejecución del juego, dejaremos marcada la opción de “Activación automática” que está más abajo. Si por el contrario la queremos activar en un determinado momento mediante

Blueprints la desactivaremos. Más adelante explico cómo activar sonidos mediante Blueprints.

6.4.2.5 Controles y Funciones de interacción con el entorno mediante Blueprints.

En esta sección hablaremos sobre las funcionalidades que se han implementado para poder interactuar con el entorno. Las funcionalidades serán el movimiento del personaje mediante teclado y ratón, o controlador de Xbox y Oculus Rift, que te permitirán moverte, mirar alrededor, saltar y un botón para interactuar con objetos tales como puertas. [12]

Movimiento del personaje

El movimiento del personaje, tanto con teclado y ratón u Oculus y mando se han implementado mediante el Blueprint de "FirstPersonCharacter" donde por defecto te viene ya una implementación básica de los movimientos del personaje incluyendo salto y aspectos de un juego *Shooter* como disparar. Esto último lo borraremos y le quitaremos al actor el arma y los brazos para que no se vean por pantalla. También añadiremos variables adicionales para modificar parámetros como la velocidad del movimiento, capacidad de salto y parámetros similares. Una vez entendemos el funcionamiento del BluePrint también podemos añadir sonidos dinámicos para que se ejecuten cuando hagamos cierta acción como por ejemplo al saltar.

A continuación, muestro el sistema de nodos de los Blueprints del "FirstPersonCharacter" con el que se ha implementado el movimiento.

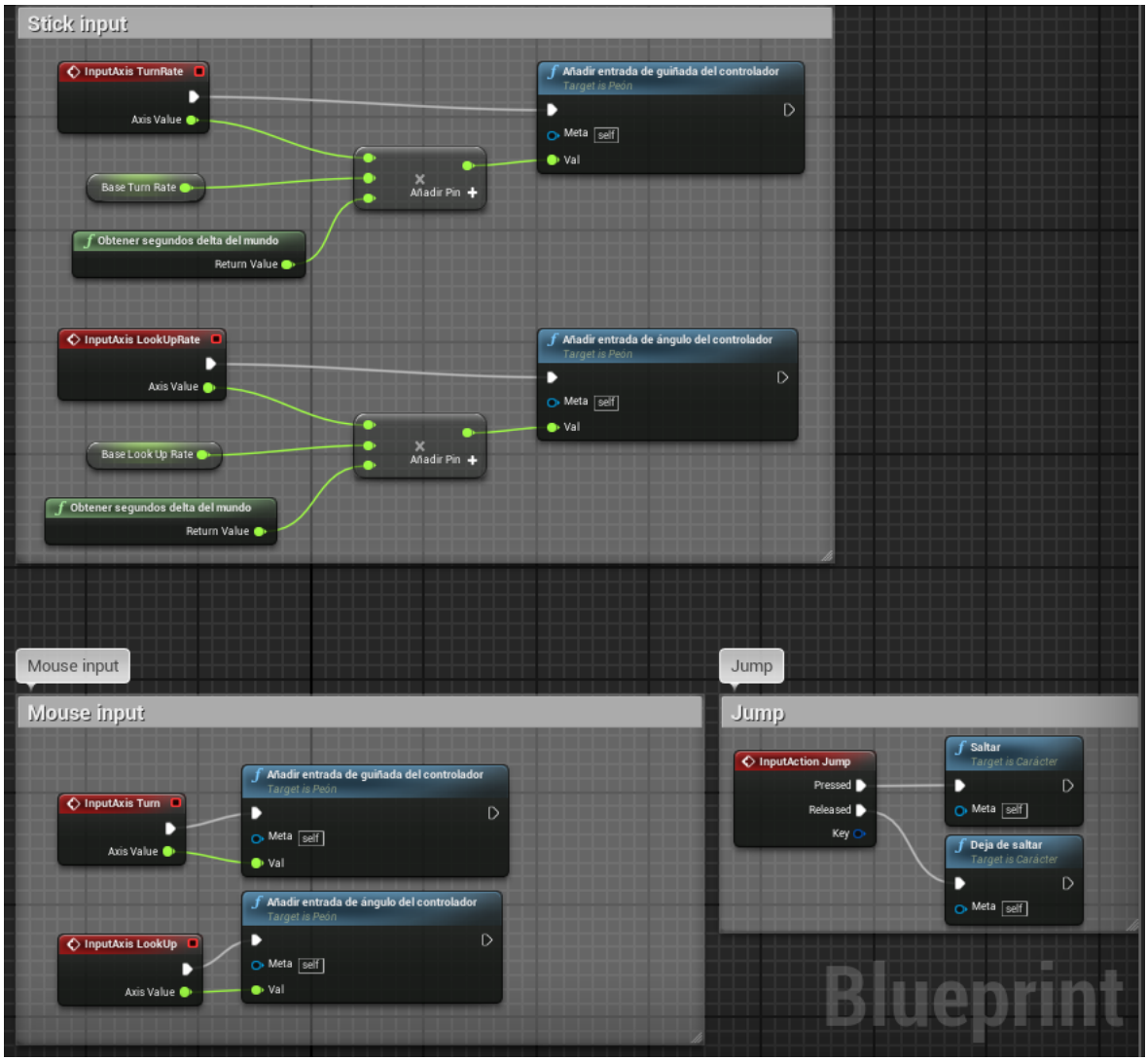


Ilustración 49. Blueprints movimiento1.

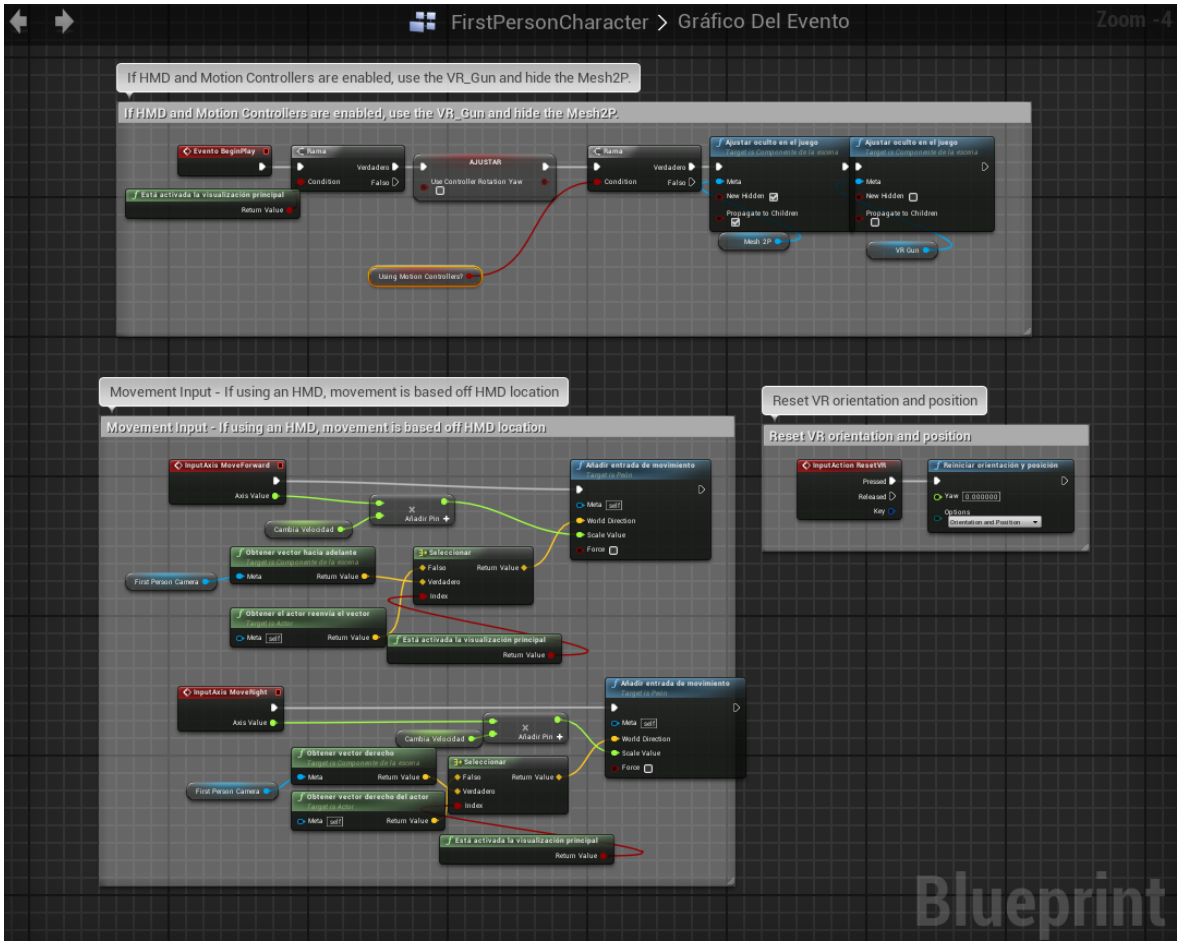


Ilustración 50. Blueprints movimiento 2.

Abrir y cerrar puertas

En el nivel se podrán abrir y cerrar algunas puertas. Para ello vamos a crear la interacción con ellas mediante Blueprints de nivel.

Lo primero que tenemos que hacer es ajustar bien el punto de pivote para que gire bien, esto es muy importante. A continuación, creamos una animación para la puerta en la opción "Cinematics" → "Add Matinee". En la ventana que se abre es donde vamos a crear la animación de la puerta. Para ello, teniendo seleccionada la puerta, hacemos clic derecho en la zona de las pistas y seleccionamos "Add new empty group" y le añadimos una nueva pista

de movimiento.

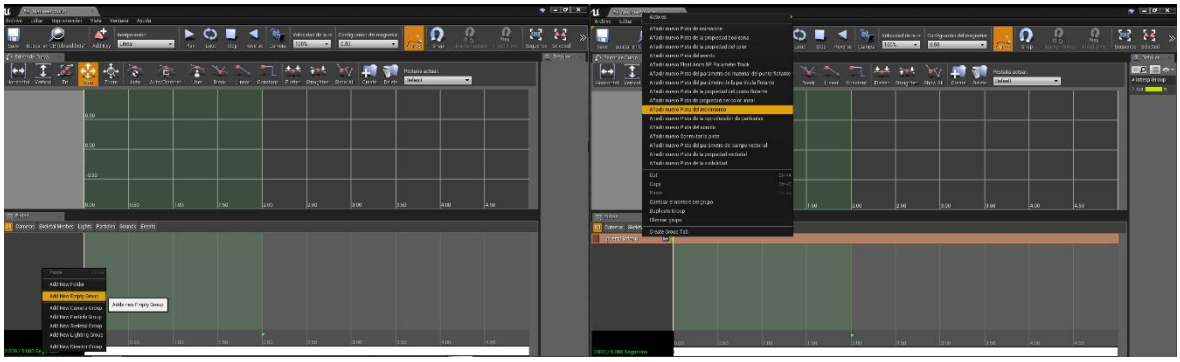


Ilustración 51. Creación de un nuevo grupo de objetos para animar

Ahora que tenemos esto, lo único que tenemos que hacer para terminar la animación será crear en el instante 0 un *key frame* con la posición inicial de la puerta, y después rotar la puerta a la posición final y crear un *key frame* en el segundo 2, que será cuando acaba la animación.

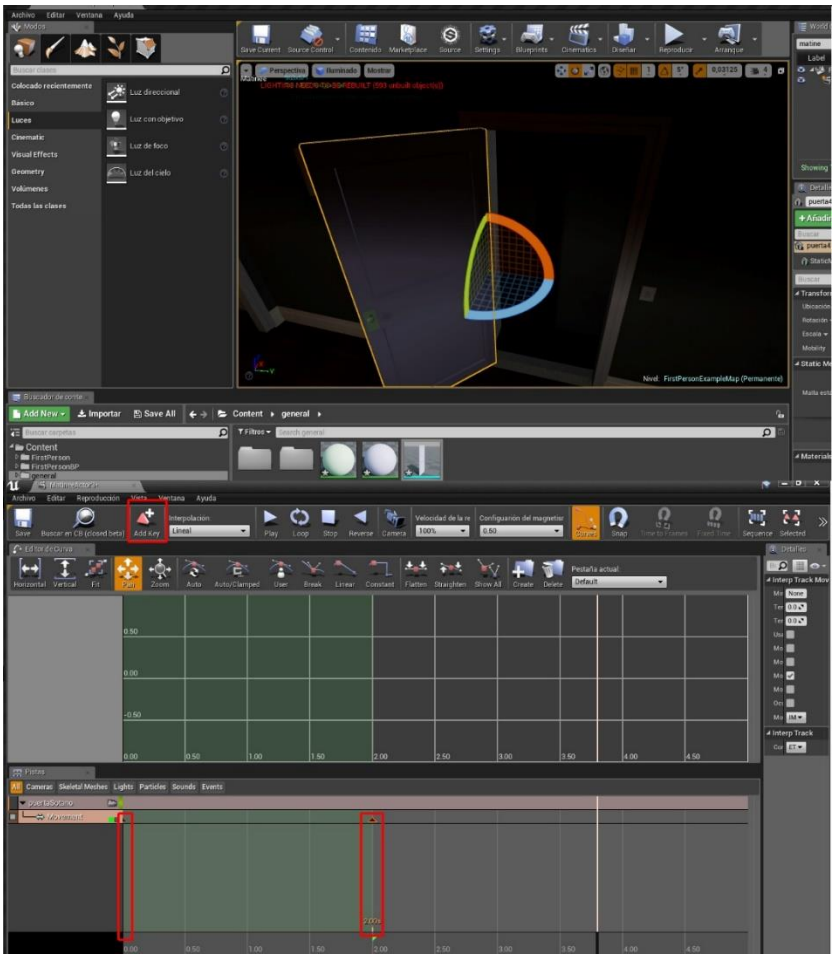


Ilustración 52. Creación de los Keyframes para el movimiento de la puerta.

Ahora hay que crear un Trigger box. Esto hace que, cuando el personaje entre dentro del volumen del Trigger, “pase algo”. Para ello lo que haremos será ponerlo al lado de la puerta y así cuando nos pongamos al lado de ella podremos interactuar.

Para poder interactuar necesitamos crear un botón que llamaremos “interacción”, para lo que nos dirigiremos a las opciones del proyecto y le asignaremos la letra F a esa acción.

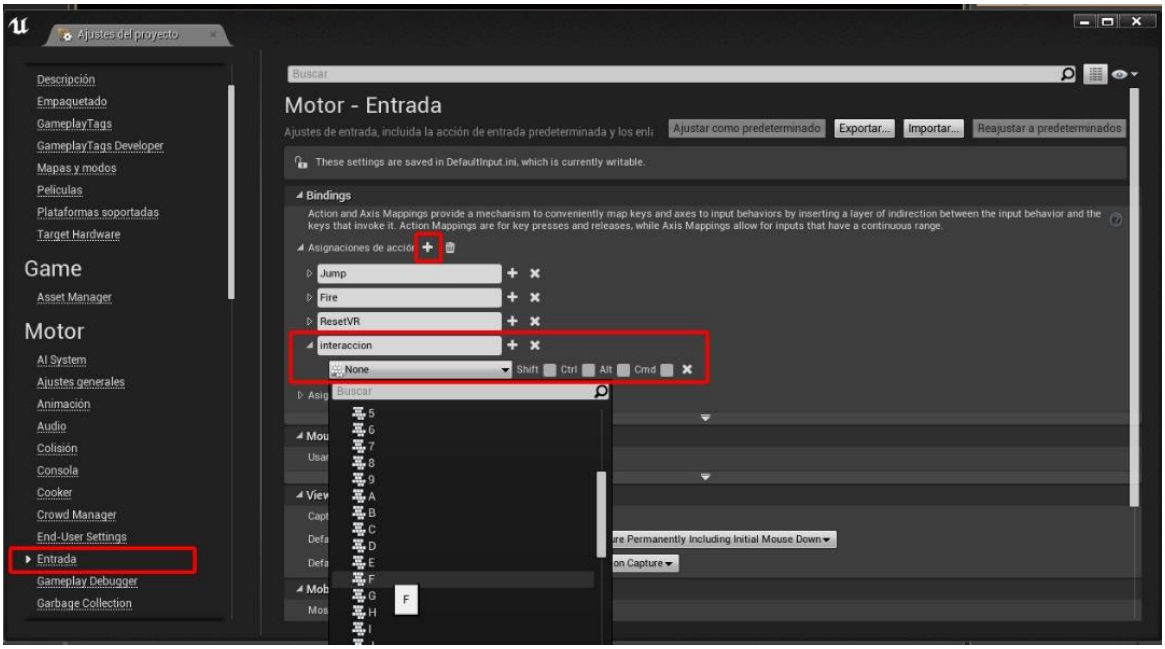


Ilustración 53. Declaración de nuevos controles para el juego

Una vez tenemos esto, podemos crear un mensaje mediante el uso de Widgets para que, cuando nos acerquemos a la puerta, nos indique que se puede abrir pulsando la tecla de interacción.

Por último, vamos a crear el BluePrint que hará que cuando entremos a la Trigger box, nos permita darle al botón de interacción y que se ejecute la animación.

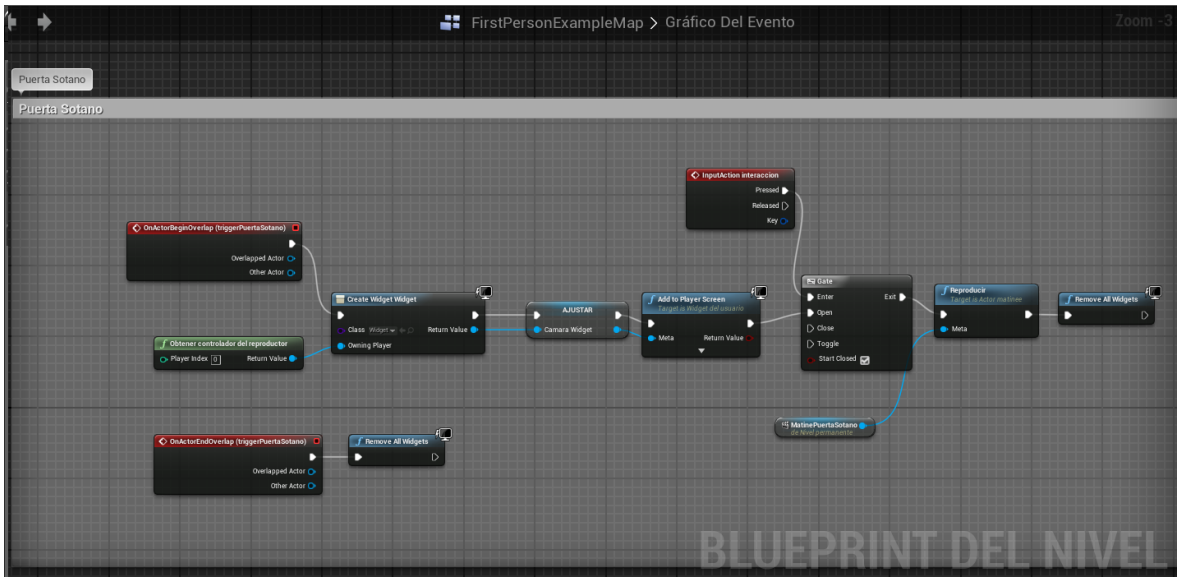


Ilustración 54. Blueprint completo de abrir y cerrar puertas

Sonidos

Hay ciertos sonidos que querremos activar en algún momento específico, por ejemplo, al abrir una puerta o al llegar a un punto donde queramos dar un susto. Para hacer esto lo que debemos hacer es usar Blueprints. Con ello podremos hacer que, mediante una tecla en el caso de abrir la puerta, se reproduzca el sonido de la puerta o que al entrar a un "Trigger volumen" se ejecute un sonido. Lo único que tenemos que hacer, por ejemplo, para que suene un cuervo al llegar a una zona, será crear un evento "OnActorBeginOverlap", una instancia del sonido y la función de reproducir audio.

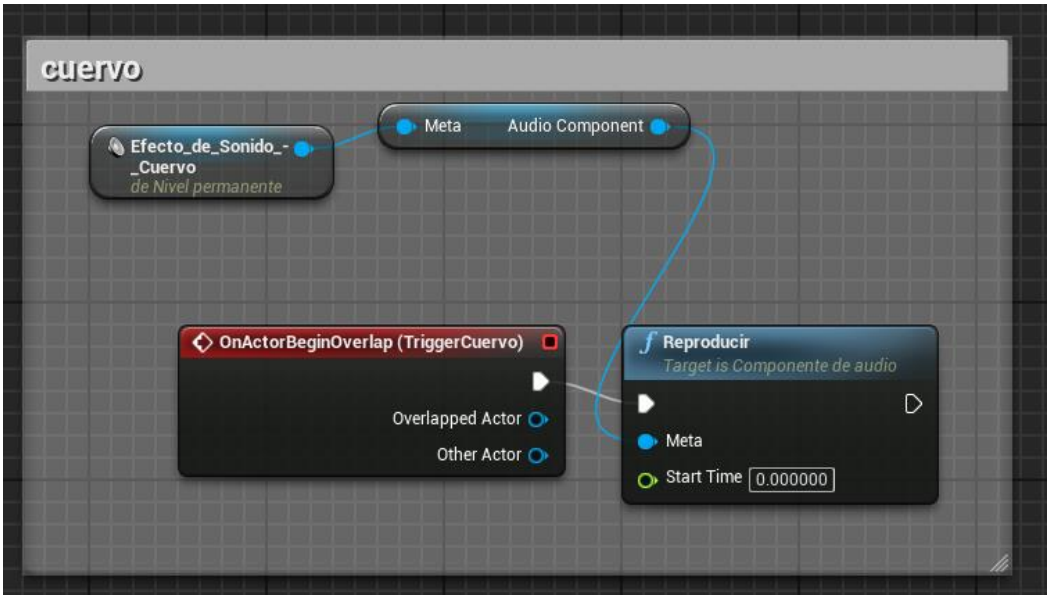
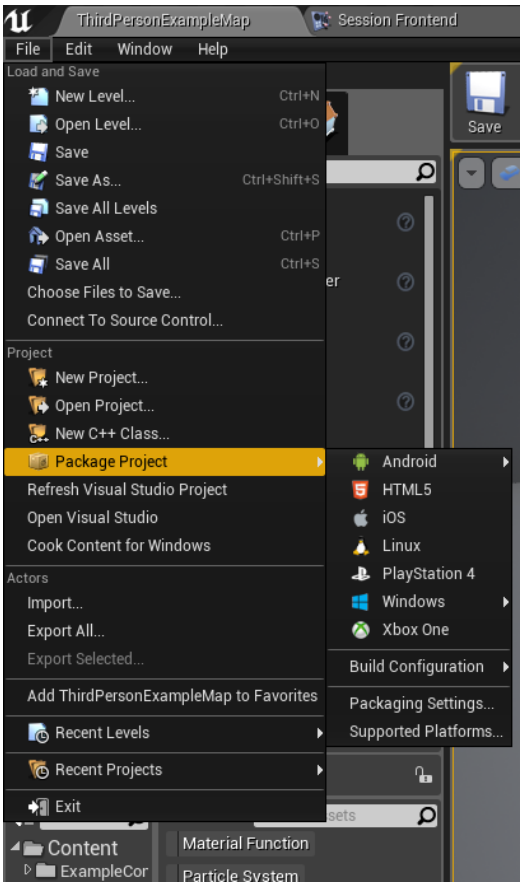


Ilustración 55. Ejemplo de Blueprint para ejecutar sonidos mediante Triggers.

6.5. Render final y empaquetado del proyecto

Una vez finalizado el proyecto, para que se vea correctamente la escena, tenemos que diseñar la escena. Esto lo que hará será generar las luces, sombras y brillos de manera correcta por toda la escena y hacer que las texturas se vean con la mayor calidad posible. Para hacer esto, lo único que tenemos que hacer será dirigirnos al menú y darle a "Diseñar" y tras un rato procesando todos los elementos se verá la escena con su aspecto final. Con el desplegable de esa misma opción, se ven los diferentes tipos de diseño que puedes hacer y con diferentes calidades.

Por último, si queremos crear el ejecutable del juego para poder usarlo en otros ordenadores, tenemos que empaquetar el proyecto. Esto lo haremos dirigiéndonos a "File" → "Package Project" y seleccionamos la plataforma en la que queramos ejecutar el juego.



6.6 Resultados finales

En este apartado se mostrarán los resultados finales obtenidos comparándolos con escenas similares de la película. En la parte de arriba de la foto se verá la película y en la parte de abajo, el juego que hemos creado.

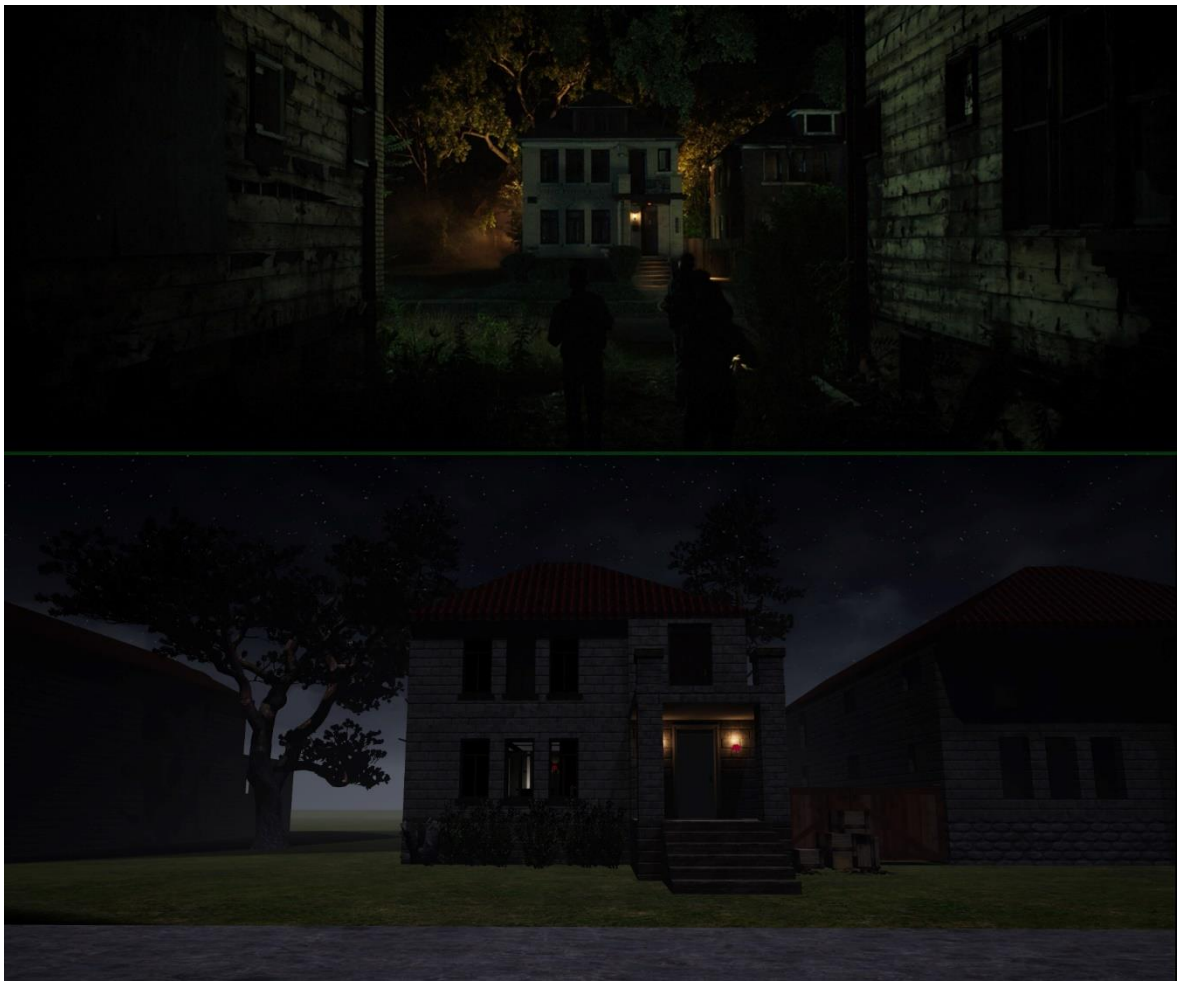


Ilustración 56. Comparación de la película con la escena de UE4 (Fachada).

Ilustración 57. Comparación de la película con la escena de UE4 (Trastero)..

Ilustración 58.Comparación de la película con la escena de UE4 (Sótano).

Ilustración 59. Comparación de la película con la escena de UE4 (Sótano2).

Ilustración 60. Comparación de la película con la escena de UE4 (Salón).

Ilustración 61. Comparación de la película con la escena de UE4 (Dormitorio).

Ilustración 62. Comparación de la película con la escena de UE4 (Pasillo)..

Ilustración 63. Comparación de la película con la escena de UE4 (Salón).

Ilustración 64. Comparación de la película con la escena de UE4 (Recibidor).

Ilustración 65. Comparación de la película con la escena de UE4 (Taller).

Ilustración 66. Comparación de la película con la escena de UE4 (Baño).

Ilustración 67. Comparación de la película con la escena de UE4 (cocina).

7.Conclusiones

Una vez finalizado el proyecto, y haber trabajado con él durante mucho tiempo, voy a hacer una serie de conclusiones sobre las técnicas utilizadas en el proceso y sobre las herramientas utilizadas.

Para empezar, podemos concluir que en cuestión de diseño de los modelados y texturizado estoy bastante contento con los resultados obtenidos y he aprendido correctamente a realizar el flujo de trabajo que se utiliza en la industria. He realizado todos los modelos y prácticamente todas las texturas y han quedado unos resultados bastante buenos. El proceso de creación ha sido muy laborioso ya que cada modelo tiene que pasar por diferentes fases (diseño, modelado, mapeado, *backing* y texturizado) para optimizarlo correctamente, pero ha merecido la pena todo el esfuerzo ya que ahora sé cómo crear desde cero entornos para videojuegos totalmente optimizados y preparados para ser usados.

Respecto a la creación del entorno, he de decir que yo nunca había trabajado con Unreal Engine ni ningún otro motor gráfico cuando empecé a hacer el proyecto. Ha sido un trabajo muy largo y que ha requerido de la búsqueda de muchísima información. La programación del videojuego, a pesar de que no ha sido el fuerte del trabajo, ya que nos hemos centrado en el diseño, puedo decir que ha sido muy fluida gracias a los Blueprints y que ha sido realmente muy intuitivo si tenemos nociones básicas de programación. Respecto a lo visual, la iluminación ha sido un trabajo muy arduo por diferentes motivos: en primer lugar, la iluminación y los materiales van ligados, por lo que cuando traemos los materiales de Substance Painter a Unreal Engine no se ven igual debido a esto, por lo que una vez creado el material había que hacer cambios en los materiales una y otra vez hasta que se viesen como debería; y las luces hay que renderizarlas cada vez que hacemos cambios en ellos, proceso que tardaba una media hora cada vez que lo hacíamos, por lo que esto ha dado bastantes dolores de cabeza.

Respecto a la semejanza con la película, a pesar de las dificultades que tiene representar un entorno sin unos planos más específicos, puedo decir que he conseguido representar el escenario de la película de manera bastante fiel y crear la ambientación de terror que pretendía al principio.

El juego se ha conseguido implementar para poder visualizarlo con gafas de realidad virtual. No obstante, a pesar de que sin ellas el juego funciona correctamente, he de decir que cuando haces uso de estas, como las propias gafas junto al sensor detectan tu altura y tu posición, hay algunos problemas con las colisiones dependiendo de quien las use.

Por último, me habría gustado añadir personajes con sus respectivas animaciones, pero suponía mucho trabajo que por falta de tiempo era imposible desarrollar, y que se queda como futura mejora. Otra propuesta de mejora, que habría estado bien implementar para móviles, sería crear la aplicación con menú y donde no seas tú el que maneja al personaje, sino que se vaya moviendo él solo por toda la casa con un recorrido preestablecido y que el usuario simplemente disfrute del recorrido con unas Cardboards[3] de realidad virtual mirando a su alrededor con ellas.

Como conclusión final, podemos decir que se han conseguido todos los objetivos que se habían propuesto en un principio; la realización de los planos de la casa de la película, creación de todos los modelos 3D y el texturizado de los mismos, llevarlo todo a UE4 y crear un entorno interactivo mediante Blueprints y crear la ambientación de terror que se pretendía y por últimos, la visualización del entorno con gafas de realidad virtual por lo que se han conseguido realizar todos los objetivos marcados.

El video con el entorno final está disponible en el siguiente enlace: https://youtu.be/wVyh73K51CU

[3] Cardboards son unas gafas de realidad virtual muy económicas compuestas por dos lentes tipo lupa y la estructura de las gafas, donde puedes introducir tu móvil y disfrutar de una experiencia VR básica.

8.Bibliografía

[1] Que es el cine:

https://es.wikipedia.org/wiki/Cine

Referencias de videojuegos de terror:

[2] Outlast: https://es.wikipedia.org/wiki/Outlast

[3] Resident Evil 7: https://es.wikipedia.org/wiki/Resident_Evil_7:_Biohazard

[4] P.T. http://es.silenthill.wikia.com/wiki/P.T.

Metodología:

[5] Kanban: https://es.wikipedia.org/wiki/Kanban_(desarrollo)

[6] Trello: https://es.gizmodo.com/como-organizar-toda-tu-vida-utilizando-trello-1684529913

Motores Gráficos

[7] Qué son: https://es.wikipedia.org/wiki/Motor_de_videojuego

Modelado 3D

[8] Qué es: https://es.wikipedia.org/wiki/Modelado_3D

Materiales y texturas

[9] Qué son: https://prezi.com/92sw3at3hgrp/materiales-y-texturas/

GDD

[10] Diseño del GDD: https://eldocumentalistaudiovisual.com/2015/02/06/documentacion-en-videojuegos-documento-de-diseno-gdd/

UE4

[11] Conceptos: Documentación oficial de UE4

[12] Blueprints: (GomVo tutorials)
https://www.youtube.com/channel/UCjj_5i2y2UcGvxtBkUYWcNg